Proceedings of Navy SBIR Neural Network Conference

June 4 -5, 1992

Proceedings of Navy

SBIR Neural Network Conference

June 4 - 5, 1992

Preface

This volume contains sixteen contributions by invited representatives to a Neural Network Applications Conference sponsored by the Navy's Small Business Innovation Research (SBIR) Program. The Office of Naval Research (ONR) was the first major governmental organization to have a program dedicated to the basic science of understanding neural networks. The program began six short years ago and quite soon it became evident that Navy-relevant applications were possible.

The past six years have shown that neural networks have progressed from being great ideas in their creators' heads to engineered applications that show results. In the beginning there was no clear winner in the Great Best Network Model Race, but the papers in this volume reveal that "backpropagation" (in its various forms) has pulled far ahead of the pack. I would suggest the following reasons: Backpropagation training is mathematically organized to produce a minimization of the mean squared aggregate error across training patterns. Applications oriented engineers and small business entrepreneurs are comfortable with LMS error techniques, and backpropagation is able to minimize the mean squared error of the training data with relatively simple mathematical proofs.

Secondly, backpropagations is a supervised training technique. This feature implies predictability, accuracy and except for the amount of training in some cases, a kind of efficiency. All that being said, let me suggest that the future may well belong to hybrid systems, recurrent networks, and/or possibly, some more "biological" variant.

Nonetheless, Mr. Vincent Schaper and the SBIR program should be congratulated on taking the important first step to support neural network technology so quickly. The topics and applications here include: data fusion, motor controllers, target recognition, signal classification, and others. All are important requirements for enhanced future war fighting ability of the Navy, and as this volume shows, all are fertile grounds for neural network applications in the commercial arena.

Joel L. Davis
ONR
Code 1142 CN

Table of Contents

Autonomous Control

Other

Introduction

The Navy SBIR Neural Network Technology Conference represents the first technology conference of its type in the Navy and DoD. It was intended to allow participants in the Navy's SBIR Program who have had progress in their SBIR projects to present technical achievements and accomplishments to representatives of the Navy's "user" or Program Management community, other DoD services, Defense Prime Contractors, Department of Transportation (DOT), and major non-defense industry in order to provide an atmosphere for transfer/transition of the technology to government and non-government users and buyers of such technology. It is the Navy's intention to continue conferences in this technology within the next two to three years and in other technologies in the interm.

SBIR Conference Attendees

* Indicates Presenter

Abbott, LDCR Gary	Naval Air Warfare Center
Akita, Dick	Naval R&D Division of NCCOSC
Baran, Robert	Naval Surface Warfare Center
Blackburn, Michael	Naval R&D Division of NCCOSC
Brase, James	Naval Supply Systems Command
Brownell, Thomas	General Electric
Buchanan, Gary	Space & Naval Warfare System
Chen, P.F.	U.S. Army Topographic Eng. Center
Cox, Chad	Accurate Automation Corp.
Cunningham, Ron	Logicon
Davis, Joel	Office of Naval Research
* Dawes, Robert	Martingale Research Corp.
Duchak, George	Space and Naval Warfare System
Elshoff, Jim	General Motors Research
Farsaie, Ali	Naval Surface Warfare Center
Ferkinhoff, Dave	Naval Undersea Warfare Center
* Fisk, Michael	Systems Engineering Associates
Fitzgerald, Raymond	General Electric
Forry, Joseph	Harry Diamond Laboratories
Franklin, Dr. Jude	PRC, Inc.
* Fuqua, Jerry	Orincon
* Gonsalves, Paul G.	Charles River Analytics, Inc.
* Greenwood, Dan	Netrologic
* Gutschow, Todd	HNC, Inc.
Harper, Ronny	Accurate Automation Corp
* Hickman, Gail	Innovative Dynamics
Honner, Nicolas	GEC Avionics
Hull, Stephen	Naval Air Systems Command
Jex, Henry	Systems Technology, Inc.
* Johnson, James	Netrologic
Johnson, Joe	Office of Advanced Technology
Kanal, Laveen	LNK Corp.
Kerr, Thomas	Lincoln Lab of MIT
Kimm, Janet	TRW
Koeing, Paul	Defense Technology Security Admin.
Kottas, Jim	Symbus Tech
Kovatch, George	U.S. Department of Transportation
* Kuperstein, Michael	Symbus Laboratories
Lau, Clifford	Office of Naval Research

Larkin, Michael	Prometheus, Inc.
Loyer, John LTCR	Naval Air Systems Command, PMA 264
* Marin, Aliana	Accurate Automation Corp.
Massey, James	Naval Civil Engineering Laboratory
* Meagher, Georgianna	Advanced Resource Div. Corp.
Metzler, Craig	Martin Marietta Aero & Naval Systems
* Moore, Andrew	Tanner Research Inc.
Morgan, David P.	Lockheed, Inc.
Mueller, Paul	Corticon, Inc.
Murray, John	Martin Marietta
McKenna, Thomas	Office of Naval Research
McLaren, Robert	Kaman Sciences Corp.
McMullen, Teresa	Office of Naval Research
* Naim, Ari	Reshet, Inc.
Nguyen, Chung	Naval Undersea Warfare Center
* Pap, Robert	Accurate Automation Corp.
Patil, Prabhaker	Ford Motor Co.
Poros, Demetrios	Jamieson Science & Engineering
Raghavan, Srinivasan	LNK Corp.
Sandel, Alex	U.S. Army Tank Automobile Command
Sansanowicz, Zisel	Naval Sea Systems Command
Schaper, Vincent	Office of Advanced Tech, Navy SBIR Program
Sherman, Porter	SIKORSKI
Shine, Jim	U.S. Army Topographic Eng. Center
Snow, Frank	SPAWAR
* Stites, Robert	Ikonix, Inc.
Stith, Robert	Marine Corp Systems Command
Tetrault, Paul	Naval Supply Systems Command
Venetsky, Larry	Naval Air Warfare Center
Wann, Mien	Naval Air Warfare Center
Whittington, Linda	Naval Supply Systems Command
Wilson, Paul	U.S. Army
Yoon, Douglas	Logicon
Zaboski, Michael	Naval Supply Systems Command

Target

Recognition

REAL-TIME OBJECT RECOGNITION USING ADAPTIVE DISTRIBUTED DETECTION TECHNIQUES

Ari Naim and Qiru Zhou[1]
Reshet Inc., 314 N. 32nd Street, Philadelphia PA 19104

Moshe Kam
ECE Department, Drexel University, Philadelphia PA 19104

ABSTRACT

A hardware realization of a data fusion system for distributed detection is described. It allows the integration of three detectors that operate at different geographical sites or employ different physical principles in making their decisions. The main objective is to process in real time large volumes of data which are received simultaneously from several remote sensors and decision makers. Specifically, the system integrates several radar stations that transmit their *target/no target* decisions to a central processor which weighs their opinions and synthesizes a final global assessment of the volume of surveillance. Once the target was detected, state estimates that are generated by the local detectors are fused at the central station to provide reliable estimates of target position and velocity. An important feature of the system is its capability to learn on-line some features of the processed data, enabling operation in nonstationary environments with only minimal a priori information. In the paper, decision and integration rules employed by the system are described, and the hardware and software designs are outlined. Performance that was measured on the physical prototype is demonstrated to agree with theoretical predictions.

1. THE GENERAL PARALLEL SENSOR FUSION ARCHITECTURE

Figure 1 presents the main components of a general distributed detection system. A group of local sensors observes a phenomenon in a surveyed volume, where each sensor 'sees' the phenomenon using a different physical principle, and at a different signal to noise ratio (SNR). Communication between sensors is limited or does not exist. On the basis of its own observations, each sensor (= local detector) forms: (i) a *target/no target* decision about the

[1]This work was supported by a NSWC SBIR grant N60921-90-C-0108

existence or non-existence of a sought object inside the volume of surveillance; and (ii) a decision about the *class* to which the object belongs.

The local decisions are transmitted to a central processor, the **Data Fusion Center** (DFC). Using the local decisions, the DFC forms a set of global target/no target decisions and target classifications, which are optimal with respect to a Bayesian -type criterion.

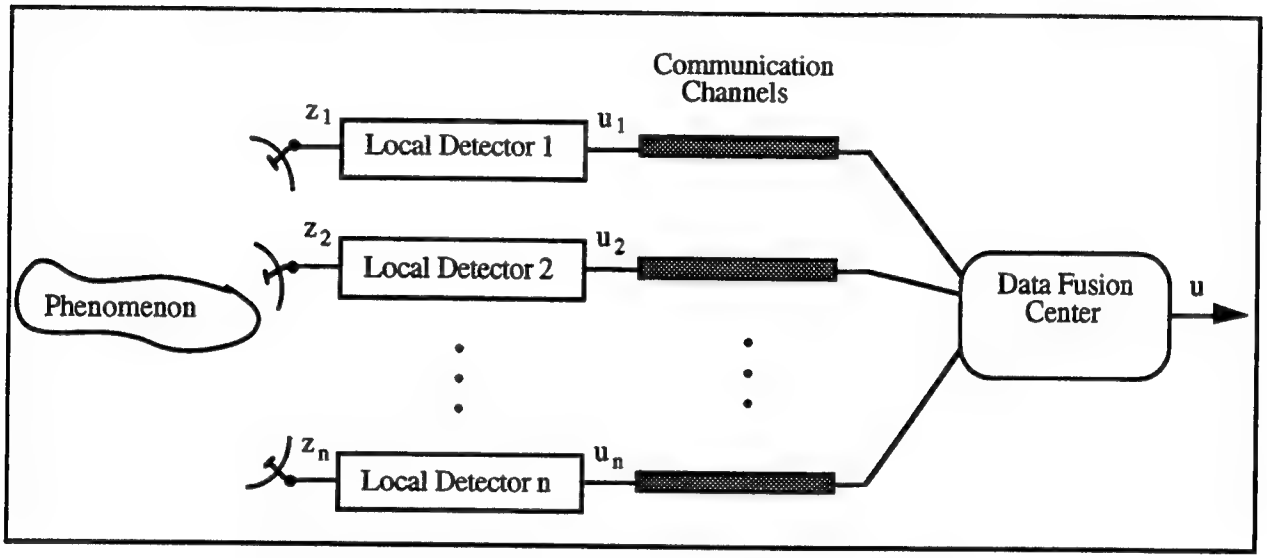


FIGURE 1: Architecture of the general distributed detection system.

The problem is generic, and a system of this kind can be used in many applications. Some of these applications are:

- Multi-sensor radar surveillance, early warning and navigation systems.
- Digital communication systems which employ diversity (in frequency, polarization or space).
- Vision systems as in aerial photography, early warning and countercrime measures.
- Parallel Computing systems which employ redundancy in order to increase reliability.

At the present time many systems with several sensors that process information from a common volume of surveillance do not use intelligent fusion. Low-rate signals are still fused by humans, and automatic fusion often employs suboptimal strategies such as a crude voting mechanisms (which provide equal weights to all decision makers), or a *winner-takes-all* strategy. The main disadvantage of the use of human operators is, of course, the high cost, the limitations on the type of signals that humans can fuse (in frequency and in complexity) and the lack of consistency and repeatability. Popular simplified automatic fusion rules suffer from degradation when compared to the optimal system. Sometimes their performance is even worse than that of the single best local decision maker. Another disadvantage is the lack of adaptability. No commercially available system has learning capabilities, in the sense that local

and global decision rules are adapting the system to the changing environment in which the sensors operate.

2. PRINCIPLES OF OPERATION - DISTRIBUTED DETECTION

Our prototype has two basic modes of operation as a distributed-detection system: a *static* mode and the *dynamic* mode. In the *static* mode no attempt is made to study the behavior of the environment in which the sensors operate; the environment is either assumed fully characterized statistically, or worst-case assumptions on the environment are made. In the *dynamic* mode, the system tries to learn on-line the statistics which are assumed known to the static system[2]. The estimated parameters are substituted in the static system's rules.

The architecture accommodates the following local decision procedures (for a complete mathematical description of these procedures see (Sage and Melsa, 1971)) :

a) Minimization of the probability of error / Bayesian risk

Each local detector tunes its internal parameters such that its decision minimizes locally the probability of error or a Bayesian risk (a weighted sum of the local probabilities of false alarm and missed detection); the DFC (given the local detectors' objective function) tries to minimize the *global* probability of error or a global Bayesian risk. By minimizing the same performance index at each site, the overall system does not necessarily achieve a global optimum. The suboptimal architecture is however easy to implement, and the learning rules are computationally efficient (unlike those of the globally-optimal architecture (e.g. Reibman and Nolte 1987)).

b) Minimax decision

Lacking information on the a priori probability of the target, each local detector makes a worst-case assumption about the environment, and tunes its parameters to minimize a Bayesian cost in the face of the least-favorable conditions.

c) Neyman-Pearson detection

Each local detector devises its decision rule so that the probability of false alarm is bounded by a specified value, and the local probability of detection is maximized. The DFC has its own bound on the probability of false alarm, and it uses the local decisions in order to maximize the *global* probability of detection. Our Neyman-Pearson version

---

[2] For a complete description of the learning rules see Kam et al. (1991) and Naim et al. (1991).

has the same bounds on the local and global probabilities of false alarm, with the result that the global probability of detection is higher than the local probabilities of detection at each sensor site (Thomopoulos et al. 1987). It is important to note that unlike the other two cases, the optimal DFC decision rule in this case is *randomized.* Deterministic decision rules at the DFC do not generally satisfyNeyman-Pearson optimality.

We assume that all local decisions are statistically conditionally independent[3], and characterize each local detector through its *probability of false alarm,* $P_{Fi} = P_r(u_i=1 \mid H_0)$, and its *probability of missed detection,* $P_{Mi} = P_r(u_i=0 \mid H_1)$. Under these conditions, Chair and Varshney (1986) have shown that it is sufficient to consider at the DFC the statistic $S(u_1, u_2, \ldots, u_n) = \sum_{i=1}^{n} u_i \log \frac{(1-P_{Mi})(1-P_{Fi})}{P_{Mi}P_{Fi}}$. The fusion rule then becomes:

$$\text{If } S(u_1, u_2, \ldots, u_n) \geq t \qquad \text{accept } H_0 \qquad (1)$$

$$\text{Otherwise} \qquad \text{accept } H_1$$[4]

The threshold t is determined by the objective function. Figure 2 demonstrates the resulting architecture for the Bayesian case, where the a_i coefficients (i = 1,2,...,n) are functions of P_{Fi} and P_{Mi}. a_0 is a function of the a priori probability $P(H_1)$ and the Bayes cost (see Sage and Melsa 1971 for their definition). The architecture is similar to that of a perceptron, which suggests the use of perceptron training rules.

[3] The case of correlated decisions has been studied by Kam et al. (1992).

[4] In the Neyman Pearson case a test of the form (1) is chosen from among two tests with different values of t.

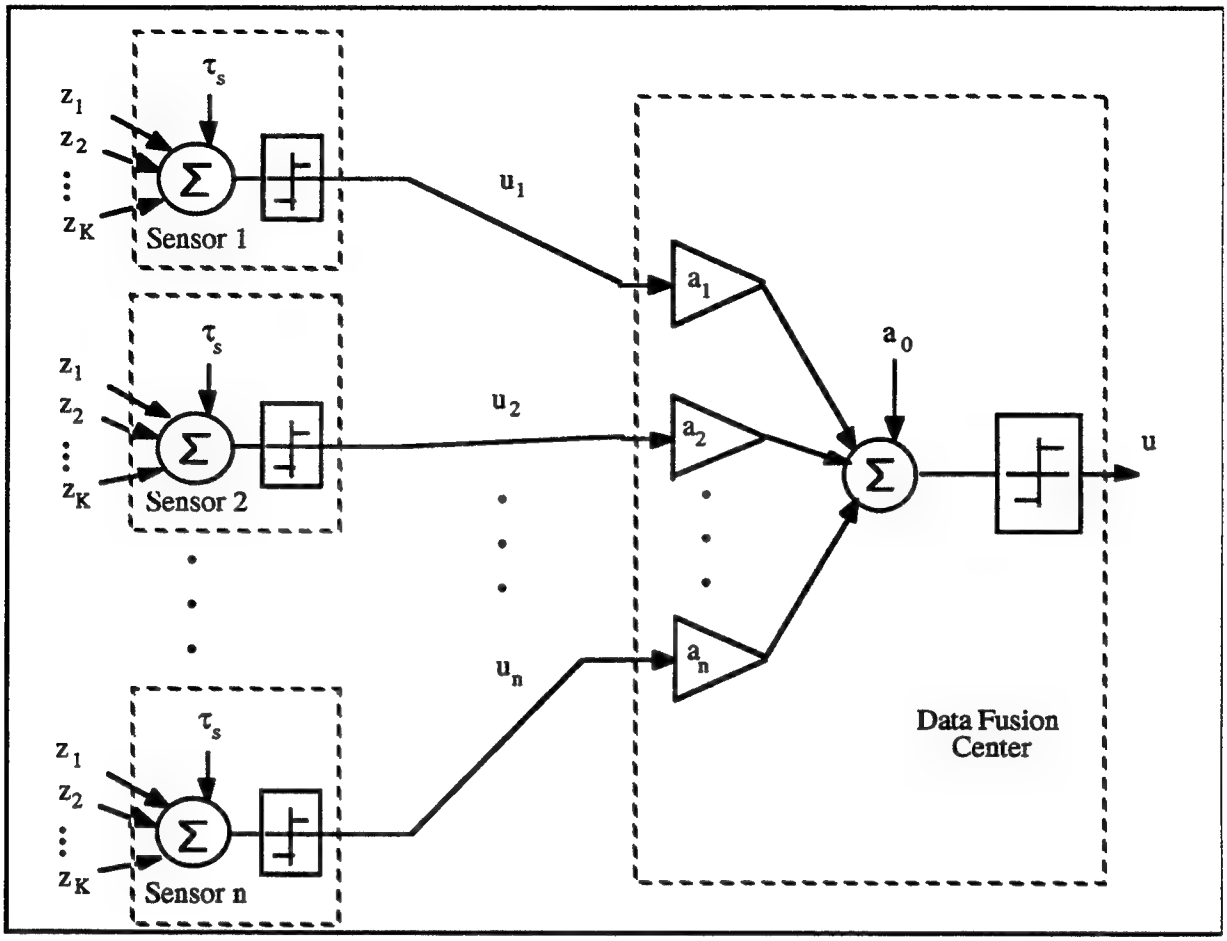


Figure 2: The detection architecture for a Bayesian objective function.

The Bayesian decision maker needs the probabilities $P(H_1)$ (= probability of a target), and P_{M_i} and P_{F_i} in order to devise the decision rules. For the minimax and Neyman-Pearson tests $P(H_1)$ is not necessary, but the local-performance probabilities are still required. When these probabilities are not known in advance, we have devised (Naim et al. 1991) on-line estimation rules to compute them. These rules are based on the theory of stochastic approximation, and are close in structure to the training rules used for sigmoidal multiperceptrons.

Example 1: On-line Learning.

To illustrate the utility of our adaptation rules, we present results for a system with five unequal detectors who do not possess the a priori target probability $P(H_1)$ and a DFC which does not know $P(H_1)$, P_{F_i}, or P_{M_i}, i=i,...,n. The system uses $P(H_1) = 0.7$, with an initial estimate of $\hat{P}_i(H_1)^{(0)} = 0.5$ by the local detectors, and $\hat{P}_C(H_1)^{(0)} = 0.5$ by the DFC. Initial estimates for P_{M_i} and P_{F_i} are 0.05, for all i=i,...,n. The local detectors operate in additive Gaussian, zero-mean white noise $N(0,\sigma_i)$, i=i,...,n, with signal-to-noise ratios (SNRs):

Local detector 1: 6.0dB
Local detector 2: 4.0dB
Local detector 3: 3.0dB
Local detector 4: 0.0dB
Local detector 5: -3.0dB

Two systems are compared: SYSTEM I - utilizes classical stochastic approximation rules for the estimation of unknown probabilities (e.g., Benveniste and Ruget (1982)); and SYSTEM II - a reduced-bias stochastic approximation system, developed by us for the project (Naim et al. 1991)

The local likelihood-ratio test after K observations by local detector i is (Sage and Melsa (1971), p.129)

$$\frac{1}{\sigma_i\sqrt{K}}\sum_{j=1}^{K} z_{ij} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{\sigma_i}{\sqrt{K}m}\ln\frac{P(H_0)}{P(H_1)} + \frac{\sqrt{K}m}{2\sigma_i} = \Gamma_i(\tau) \qquad (2)$$

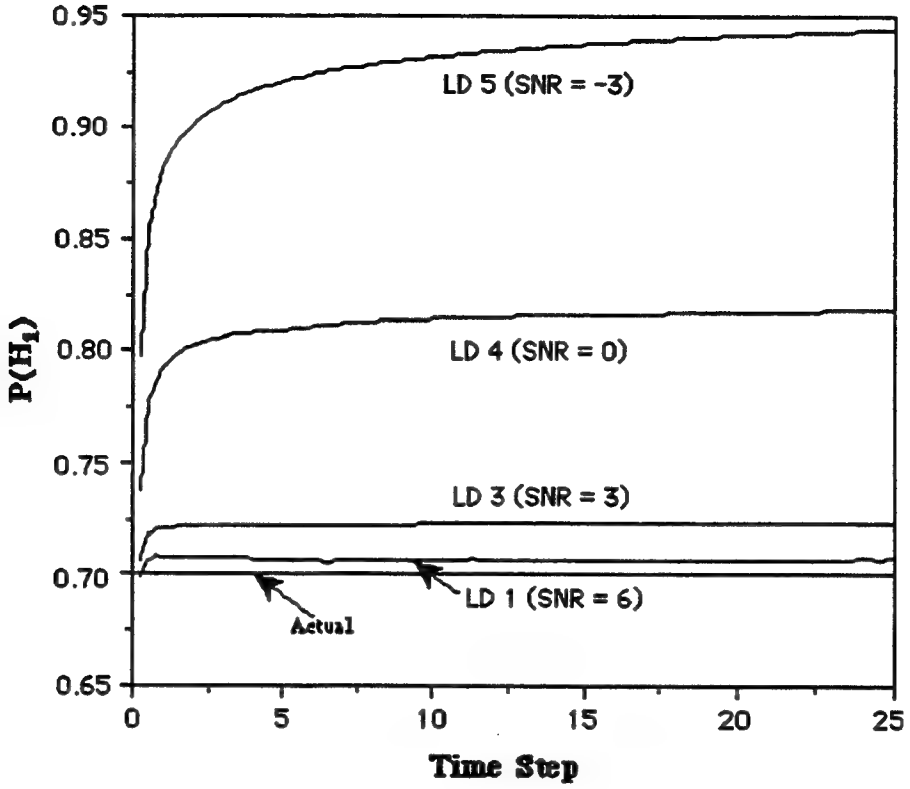
where m is the strength of the signal under H_1 and σ_i is the standard deviation of the noise site i. The local probabilities of missed detection and false alarm are

$$P_{F_i} = \int_{\Gamma_i}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} d\xi \qquad (3a)$$

$$P_{M_i} = \int_{-\infty}^{\Gamma_i - \frac{\sqrt{K}}{\sigma_i}} \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} d\xi \qquad (3b)$$

Case A - Local-detector $P(H_1)$ estimates

Figures 3a and 3b show estimates of the target probability $P(H_1)$ by the local detectors of SYSTEM I and SYSTEM II, with SNRs of -3dB, 0dB, 3dB and 6dB. Figure 3a shows the (biased) estimates with classical stochastic approximation; figure 3b shows the reduced-bias estimates which we have devised. Clearly, the use of the reduced-bias rules allows for a considerably better estimate of $P(H_1)$ in lower signal-to-noise ratios.



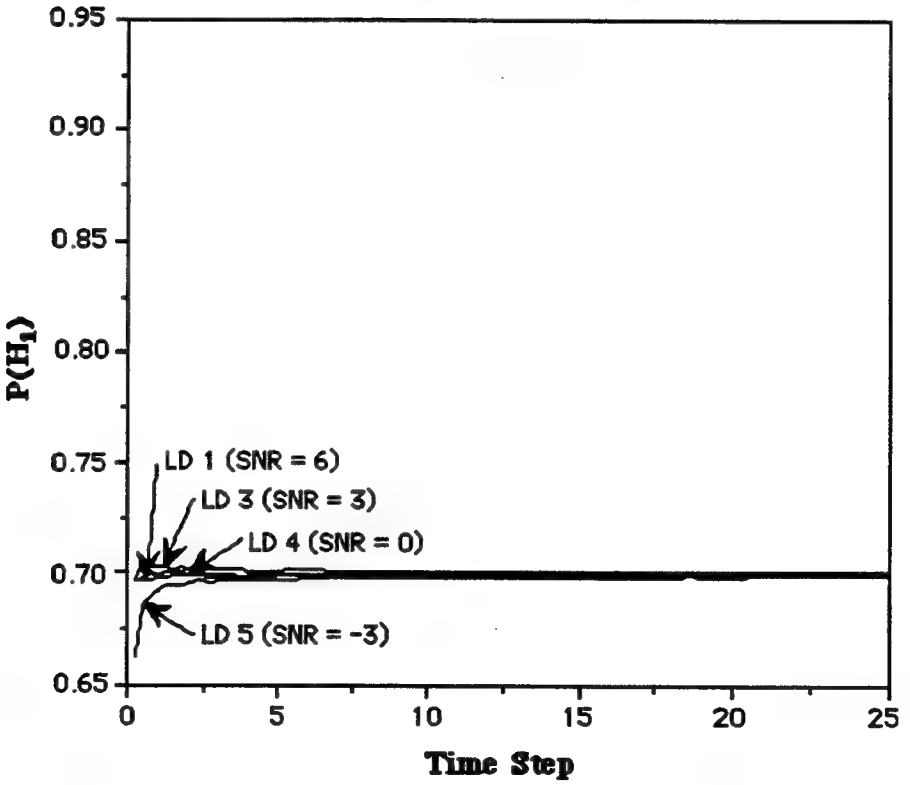


Figure 3a (top) is a comparison of $P(H_1)$ estimation by the local detectors using standard stochastic approximation rules; figure 3b (bottom) is the same comparison using the adaptive stochastic approximation rules that we have developed.

Case B - Performance of the distributed Bayesian detection architecture

In figure 4a probabilities of error are shown for local detectors 1, 3 and 5 in SYSTEM I, which uses the uncompensated estimation rule. These probabilities are compared to the exact values obtained for an optimal system that knows $P(H_1)$. In figure 4b the same comparison is made, but for the local detectors of SYSTEM II which use the reduced-bias rules. A clear improvement in local detector performance is obtained by using the bias reduction procedure (SYSTEM II). In figure 5 we compare the central (i.e., DFC) probabilities of error for systems I and II to the central probabilities of error obtained with the exact $P(H_1)$, P_{F_i} and P_{M_i}. The improvement achieved by our rules is evident.

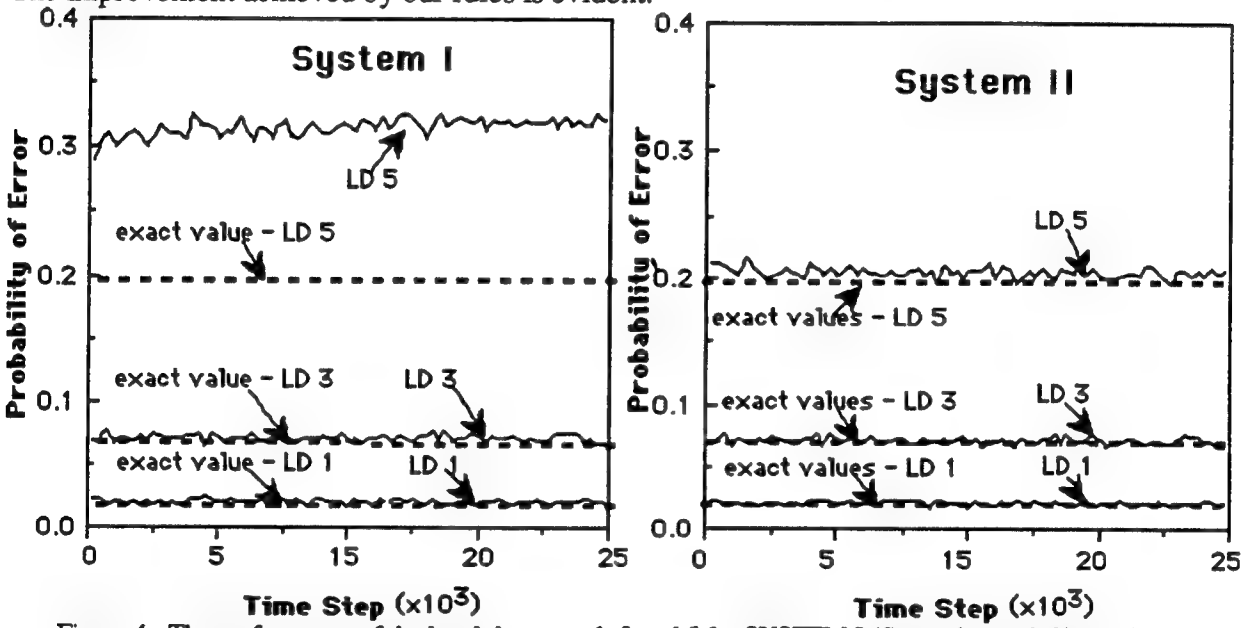


Figure 4: The performance of the local detectors 1, 3 and 5 for SYSTEM I (figure 4a, top left) and for SYSTEM II (figure 4b, top right) are illustrated. The straight dotted lines designate the exact optimal values of the probability of error.

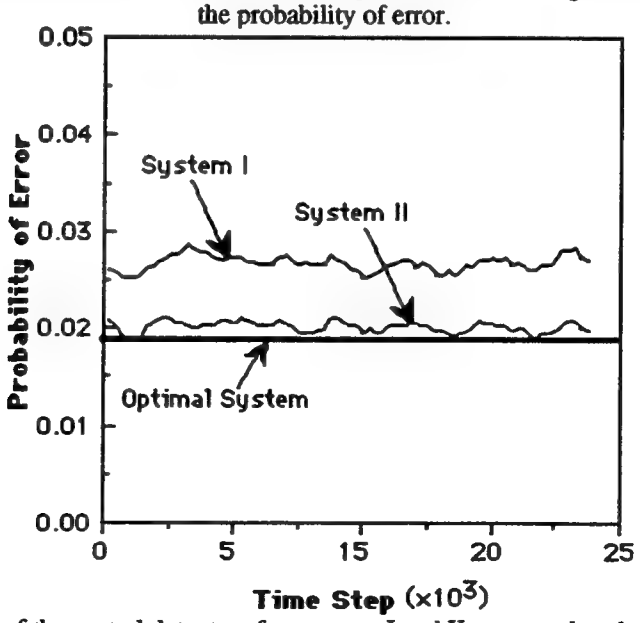


Fig. 5. The performance of the central detectors for systems I and II compared to the exact solution (given by the straight line).

Example 2: Minimax.

To illustrate the advantage of the integrated system over any single sensor, we analyze the performance of a three sensor system with local and global minimax strategy. Both the local decision makers and the DFC have the same Bayesian cost factors C_{01}=3 (cost for missed detection), C_{10}=1 (cost for false alarm), C_{00}=C_{11}=0 (cost for correct detection or correct dismissal). The local detectors operate in additive Gaussian white noise with signal to noise ratios 4dB, 5dB and 6dB for local detector 1, 2 and 3, respectively. The average Bayes cost curves are shown as figure 6. Each local detector and the DFC operate at the minimax point. The table shows the worst-case assumed target probability, and the corresponding decision thresholds:

Local Processor i	$P_i(H_0)^*$	$\log\tau_i^*$
1	0.613228	-0.637711
2	0.603563	-0.678278
3	0.593941	-0.71833
DFC	0.4	2.0

It is instructive to note that the worst case average cost of the DFC is significantly lower than the average cost incurred by each of the local detectors.

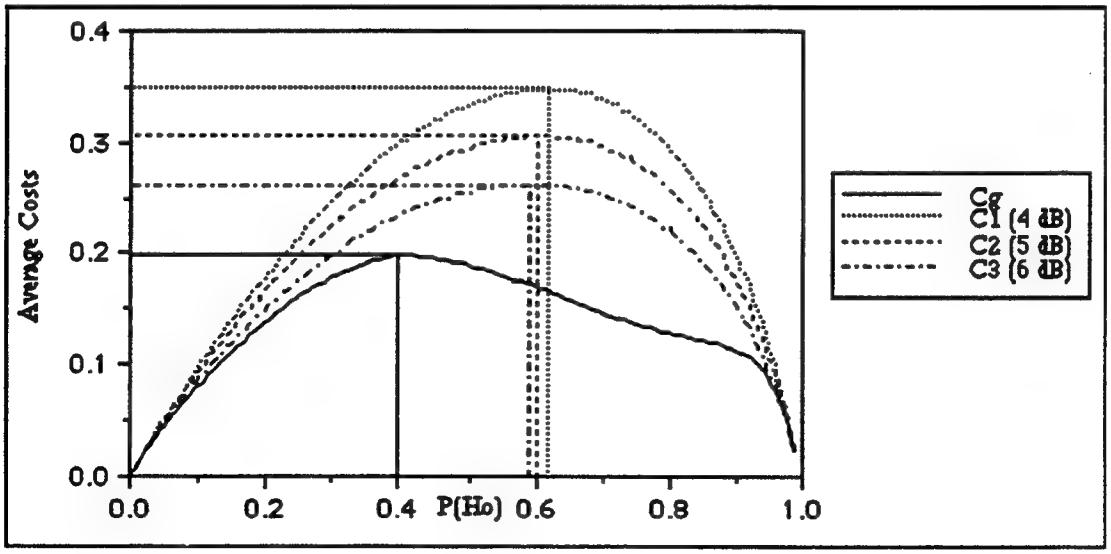


Figure 6: Bayes costs $\overline{C}$ versus $P(H_0)$ for C_{01}=3, C_{10}=1, C_{00}=C_{11}=0

3. REALIZATION

3.1 Basic Hardware Realization

The next two charts describe the basic hardware configuration used for the physical prototype. The local detectors are based on Ampro's little board 286 single-board computer. The data fusion center is based on INTEL system 120 under iRMX. The charts provide details of the internal structure of the hardware prototype for both types of detectors and show the various ports for communication with external sensors and various interconnection *local sensor - DFC* communications. At the present time the system is tested with a simulation of the environment which resides on the DFC, but the serial communication port of the local detectors allows communication with external sensors and decision makers.

The general specifications for processing rates are:

Target rate:	8000 bits per second
Number of local observations per decision:	1 - 1000
Processing rate:	500 measurements per second
Rate of DFC-local detector communications:	8000 bits per second
Rate of DFC decision making:	500 global decisions per second

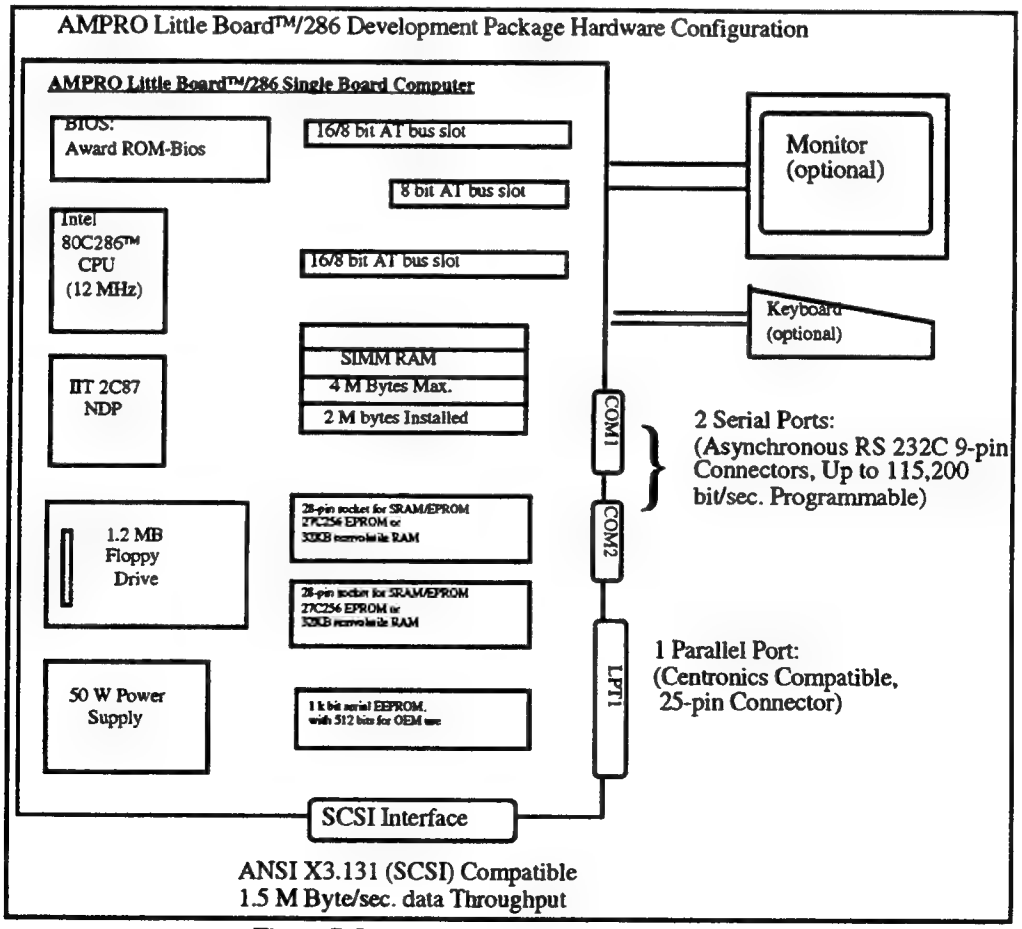


Figure 7: Local processor hardware configuration.

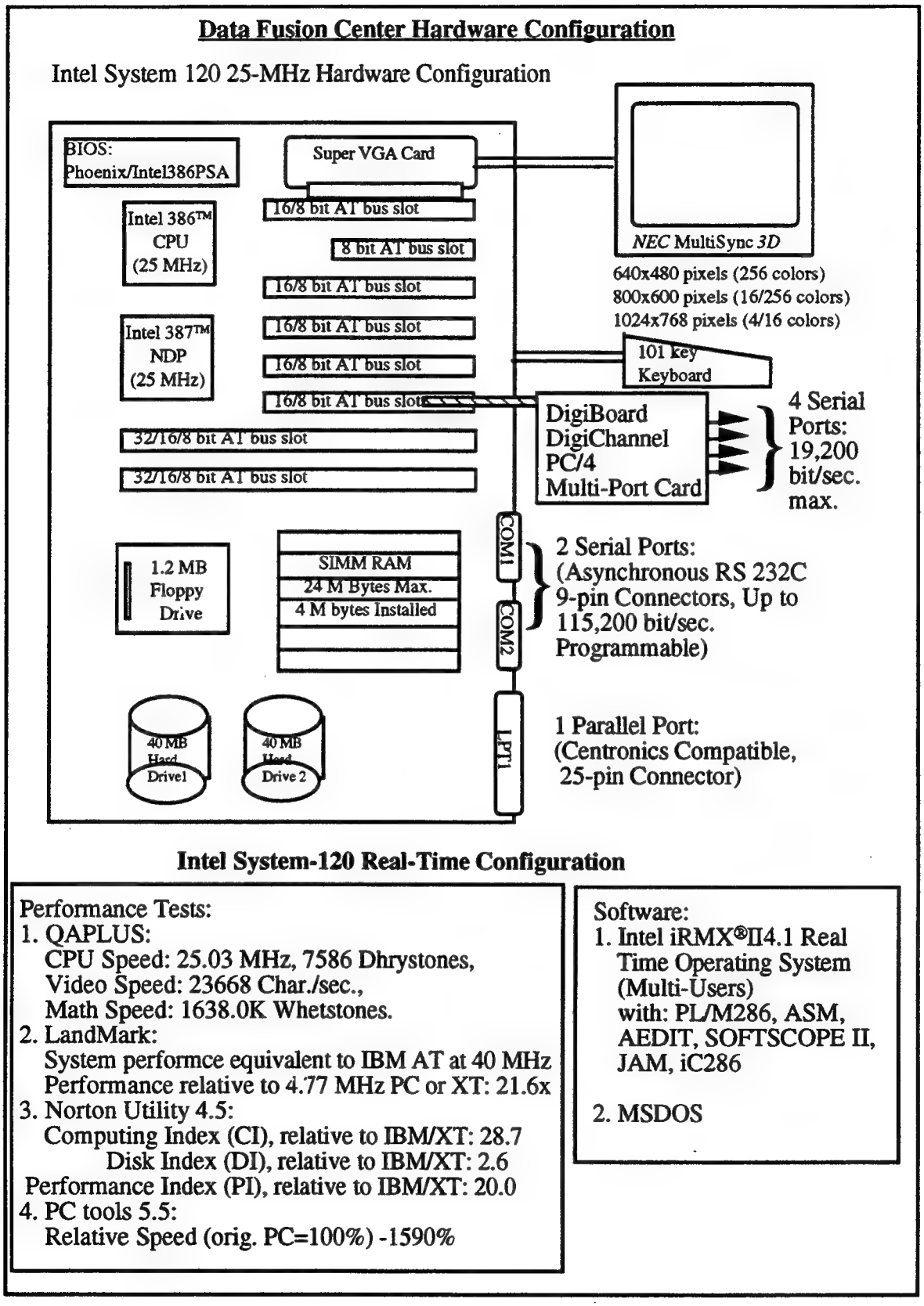


Figure 8: Data fusion center hardware configuration.

3.2 Inter-process Relation and Scheduling

The distributed data fusion system is designed to allow parallel operation for the local decision makers and the DFC, so as to take advantage of the distributed multi-processing capabilities. The inter-process relation of the LDs and the DFC is in a master-slave mode, i.e., a local processor always waits for a command from the DFC to execute an "atom" (inseparable) operation; then it becomes idle until the next command from the DFC. The master-slave operation mode simplifies system communication control signals, and system logic design. It also makes the local processor operation predictable for real-time design.

In the present configuration, the DFC supplies simulated sensor signals to all local processors. The DFC - LDs inter-process scheduling is shown as Table 1. It is a general structure valid for detection, target tracking and multiple-result packet transmission operation.

Table 1. Distributed Data Fusion System LPs-DFC Inter-Process Scheduling

	Step 0	Step 1	Step 2	•••	Step K-1	Step K
DFC:	**Send** GO command to let all LPs take the **1st** pack of sensor signal, then **wait** all LPs done, **send** X_U to LPs, **receive** LPs' results.	**Send** GO to let all LPs take the **2nd** pack of sensor signal, then do the **1st** step data fusion processing, after it done, **wait** all LPs done, **send** X_U to LPs, **receive** LPs' results.	**Send** GO to let all LPs take the **3rd** pack of sensor signal, then do the **2nd** step data fusion processing, after it done, **wait** all LPs done, **send** X_U to LPs, **receive** LPs' results.	•••	**Send** GO to let all LPs take the **Kth** pack of sensor signal, then do the **(K-1)th** step data fusion processing, after it done, **wait** all LPs done, **send** X_U to LPs, **receive** LPs' results	Do the **Kth** step data fusion processing, then stop. **The system is done.**
All the LPs:	**Receive** a sensor signal pack after **received** the DFC GO command,then do the **1st** local processing, **send** DONE to the DFC, after **received** X_U from the DFC **send** the result to the DFC.	**Receive** a sensor signal pack after **received** the DFC GO command,then do the **2nd** local processing, send DONE to the DFC, after **received** X_U from the DFC **send** the result to the DFC.	**Receive** a sensor signal pack after **received** the DFC GO command,then do the **3rd** local processing, send DONE to the DFC, after **received** X_U from the DFC **send** the result to the DFC.	•••	**Receive** a sensor signal pack after **received** the DFC GO command,then do the **Kth** local processing, send DONE to the DFC, after **received** X_U from the DFC **send** the result to the DFC.	**No operation**

First, the DFC sends operation parameters to the LDs, and then the DFC and the LDs initialize their variables and allocate memory space for operation. At *step 0*, the DFC sends a GO command, followed by a signal packet to all LDs, then waits for the LDs to finish local processing. Meanwhile, all LDs receive sensor signals and process them in parallel. They will send a DONE signal to the DFC once local decisions have been made. When the DFC receives all the DONE signals from the LDs, it sends (a X_U) command to all LDs to ask them to send their results back. The LDs send their local processing results to the DFC after receiving the X_U command. The system moves to *step 1* once the DFC received all LDs' results. In *step 2*, the DFC first sends GO command to the LDs, followed by the second signal packet to all LDs. It then begins the processing of the first local-result set. Concurrently, the LDs receive sensor signals for the next decision making. This structure allow the DFC and the LDs to work in parallel: the DFC is at the k^{th} step while the LDs are at step k+1.

3.3 DFC and Local Software Structure Design

The DFC and local-detector software modules are designed together to implement the inter-process scheduling described in Table 1. In this section, we describe the general design for both detection and target tracking, with multiple data point packet transmission capability.

The local program flowchart is shown in figure 9. There is an initialization block and "atom" operation blocks. The initialization block initializes the local system memory for the program, initializes communication ports for receiving sensor signals and for *DFC - local detector* communication. It also assigns a unique local identification number to the process. This identification flag is for *DFC - local detector* communication purposes.

There are eight commands from the DFC to start a local "atom" operation in the program. An "atom" operation has to be finished before the local system accepts another command from the DFC. These commands are:

1. FLAG: The DFC asks the local system to return its status flag. This command is for diagnostic and fault-detection purposes.

2. EXIT: The DFC asks the local system to exit the target-detection or tracking process back to its system control level. It will release program memory allocation and finish other clean-up jobs. The command is for diagnostic and error handling purposes. There is no return signal to the DFC.

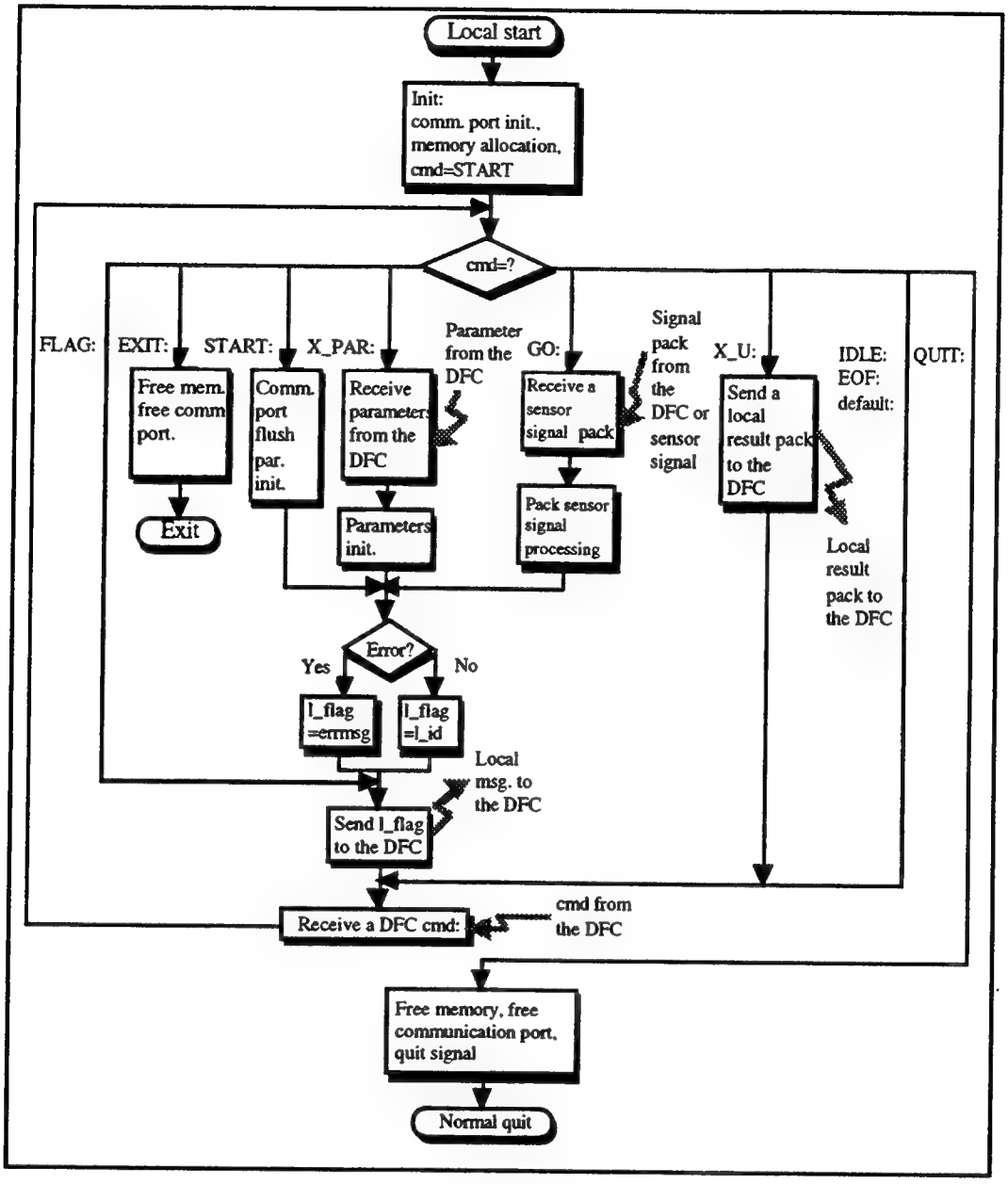


Figure 9: Local Process Flow Chart.

3. START: The DFC asks the local system to initialize for a new execution. This operation cleans up communication ports and local operation variables and parameters, such as counters, statistics registers, etc. It is used to start a new simulation or a new real-time operation without storing the previous system state. If the operation succeeds, the local system will return its local identification as a success flag to the DFC; otherwise, an error message will be returned.

4. X_PAR: The DFC asks the local detector to receive a set of operating parameters from it. The local system will receive the parameter set, then initialize its signal processing and

communication parameters. This operation will return the local identification as the success flag or an error message.

5. GO: The DFC asks the local detector to send its decisions or estimates to the DFC for global integration. Again, the operation will return the local identification as the success flag or an error message.

6. X_U: The DFC asks the local detector to send its current processing result to the DFC. There is no return flag from the local system.

7. IDLE: The local system will be idle, waiting for the next DFC command. This command can be sent by the DFC or be issued by the local detector itself after an operation. There is no return flag.

8. QUIT: The DFC asks the local system to synthesize a decision or estimate a parameter on the basis of existing data. This operation will thenrelease program memory, communication ports and stop the computations.

The DFC flowchart is shown as Figure 10. It is the visualization of Table 1.

All of the DFC and local software modules for detection and tracking are written in ANSI C code. We developed a numerical library to handle most of the numerical tasks in the algorithms, which is in the ANSI standard library format. Machine- or environment-dependent codes (such as display and communication control) are in separate modules or function libraries. Some low-level display and communication-control functions are written or in assembly language (PL/M at Intel iRMX II) for compactness and high speed.

The human user interface of the DFC is developed by using a text window interface library, which allows to write the software in "event driven" structure. This structure is akin to Microsoft windows and the X-windows system, which makes it compatible with the popular GUI environment.

The communication modules of the system are compatible with fast, multi-access local area network (LAN) hardware and related protocols. They can also be adapted to the internet network TCP/IP network protocols for long-range system operation.

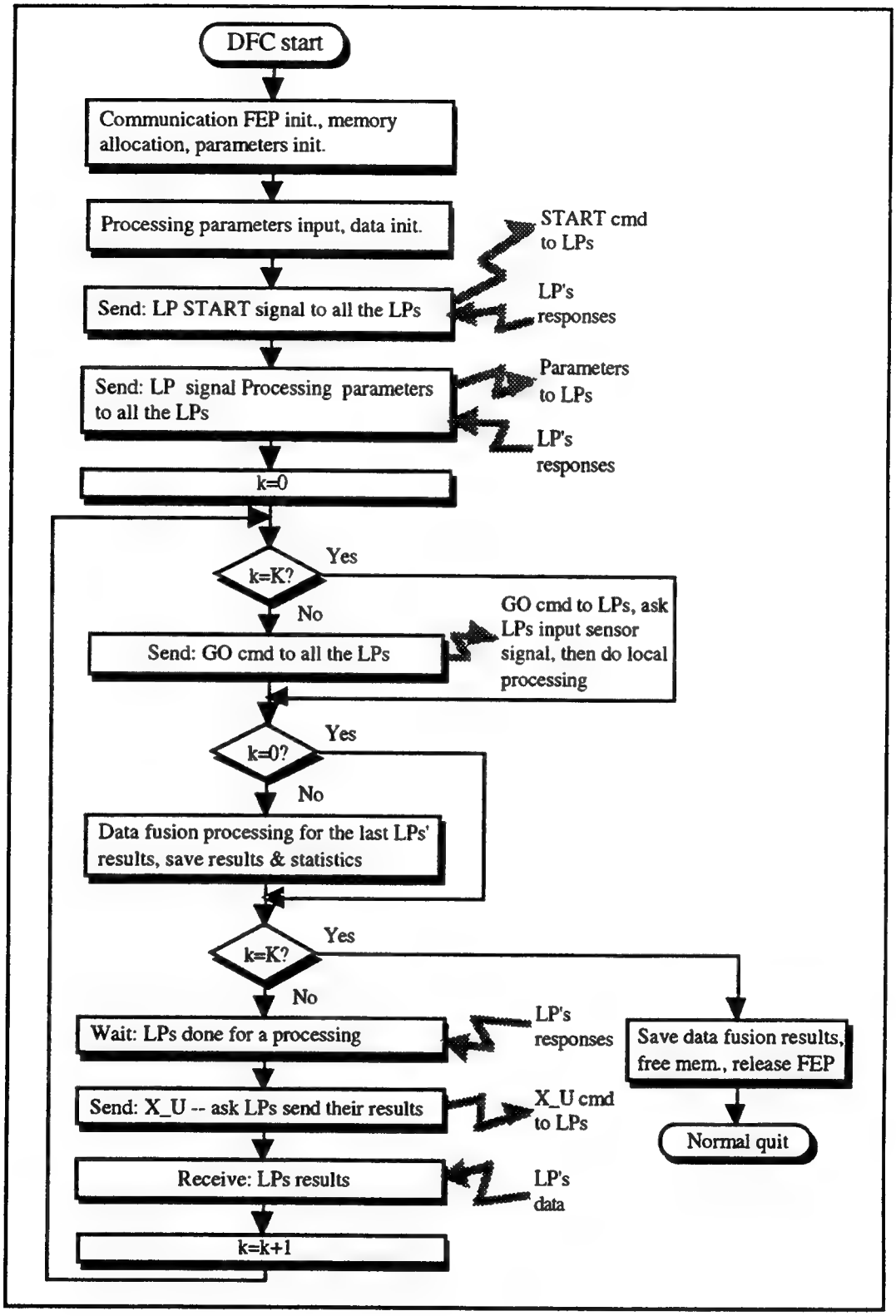


Figure 10: DFC Process Flow Chart.

4. A SAMPLE OF RESULTS FROM THE SYSTEM

4.1 Receiver Operating Characteristics

In figure 11 the theoretical and actual (prototype measured) receiver operating characteristics (ROCs) for a three-sensor system are shown. Each sensor operates in additive Gaussian noise at signal-to-noise ratio of 9 dB. The abscissa is the probability of false alarm for the (local or global) detector, and the ordinate is the probability of detection. The bottom curve is the performance of the local detectors (they are assumed identical in SNR) and the upper curve is the performance of the global system. The theoretical predictions and the actual performance are very close.

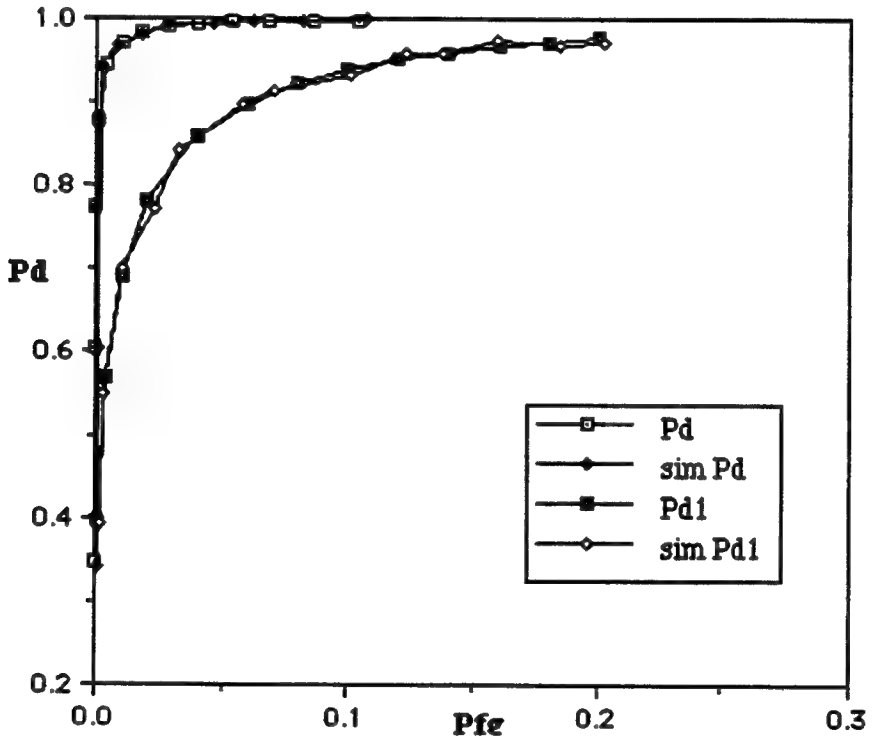


Figure 11: Comparison between theoretical and measured ROC curves for a 3-detector system designed for the Neyman-Pearson criterion (SNR=9 dB).

4.2 Estimation of Target Probability

We simulate a non-stationary environment here by changing the no-target probability $P(H_0)$. Simulation parameters are:

Target: $P(H_0)$=0.4 for the first 100 steps, $P(H_0)$=0.1 for step 100 to step 150, and $P(H_0)$=0.3 for the last 100 steps.

Minimum probability of error criteria used at local detectors and DFC.

Local observation signal-to-noise ratios (SNRs): 4 dB for LD 1; 5 dB for LD 2; 6 dB for LD 3

Local observations per decision: K=4

Tracking factors[5]: Locals α_i=0.001, i=1, 2, 3, the DFC $\beta=\beta_i$=0.001, i=1, 2, 3

Local decisions per transmission to DFC: N_p=6

Data points in graphs: 250

Samples per data point: 250

Since each of the data points is an average of 250 single decisions, the total number of global decisions made in this test is 250x250=625,000. The speed is about 42 global decisions per second.The total operation time is 24.8 minutes. Results are shown in figure 12 where both target a priori probability (solid lines) and its approximation (broken lines) by the DFC are shown.

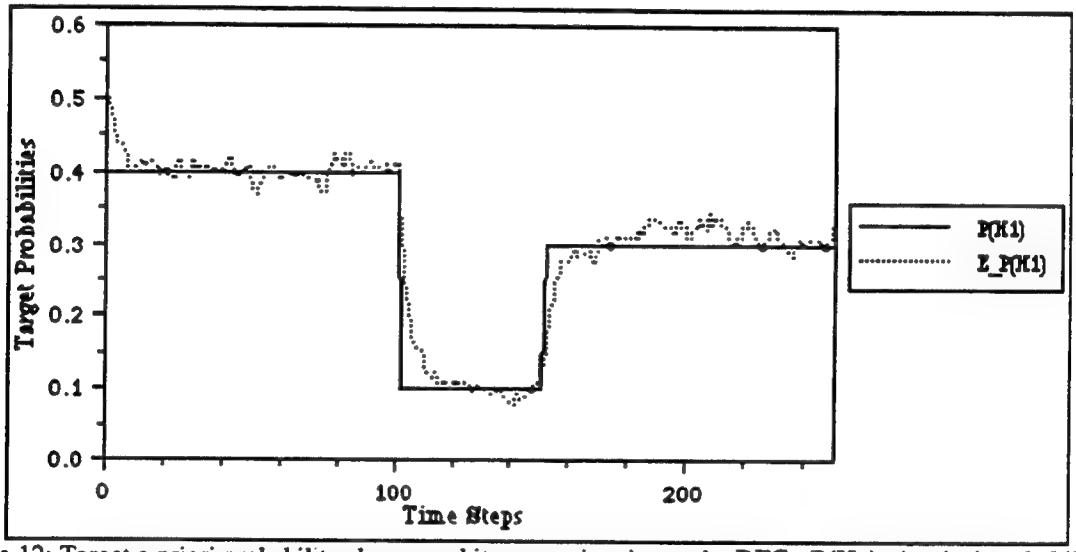


Figure 12: Target a priori probability change and its approximation at the DFC. $P(H_0)$: A priori probability in a non-stationary environment; $E\_P(H_0)$: the DFC stochastic approximation of $P(H_0)$.

The figure demonstrates that for the given learning constants and approximation rules, the a priori probability can be tracked well within 10 learning steps (k=10x250) and with small steady state variance $\mathrm{var}\{\hat{P}_0\}$.

5. CONCLUSION

The basic hardware architecture for a distributed detection architecture was presented. The detection hardware, which operates in a Bayesian framework, demonstrates close proximity to theoretical predictions and has the ability to perform with little a priori information

[5] Learning factors which determine speed of adaptation and steady state errors. For exact definitions see Kam et al. 1991.

and in nonstationary environments. Efficient learning algorithm allow tracking of slowly changing target environment, and adaptation to jump changes in sensor quality (e.g. adaptation to discrete failures). An accompanying module which provides target state estimation is also part of this system, and together these two modules can operate to acquire a target and track it.

6. REFERENCES

Bar Shalom, Y., Fortmann, T.E. (1988): *Tracking and Data Association*, Academic Press, 1988.

Benveniste, A., Ruget, G. (1982): " A Measure of the Tracking Capability of Recursive Stochastic Algorithms with Constant Gains," *IEEE Transactions on Automatic Control*, Vol. 27, No. 3, pp. 639-649.

Chair, Z., Varshney, P.K. (1986): "Optimal Data Fusion in Multiple Sensor Detection Systems," *IEEE Transactions on Aerospace and Electronic Systems*, Volume AES-22, number 1, pp. 98-101.

Kam, M., Naim, A., Chang, W. (1991): "On-Line Estimation of Probabilities for Bayesian Distributed Detection Systems in Non-Stationary Environments,"*International Federation of Automatic Control International Symposium on Distributed Intelligence Systems*, Arlington, VA, pp. 242-247.

Kam, M., Zhu, Q., Gray, W.S. (1992): "Optimal Data Fusion of Correlated Local Decisions in Multiple-Sensor Detection Systems," *IEEE Transactions on Aerospace and Electronic Systems*, (July 1992) Vol. 28.

Naim, A., Kam, M. , Farsaie, A. (1991): "On-Line Estimation of Probabilities for Bayesian Distributed Detection," *International Federation of Automatic Control International Symposium on Distributed Intelligence Systems*, Arlington, VA, pp. 235-241.

Reibman A.R. and Nolte L.W., (1987): "Design and Performance Comparison of Distributed Detection Networks," *IEEE Transactions on Aerospace and Electronics Systems*, AES-23 (Nov. 1987), 789-797.

Sage, A.P., Melsa, J.L. (1971) *Estimation Theory*, New York, NY: McGraw Hill.

Tenney R.R. and Sandell N.R., Jr. (1981): "Distributed Detection Networks," *IEEE Transactions on Aerospace and Electronics Systems*, AES-17 (July 1981), 501-510.

Thomopoulos, S.C.A., Viswanathan, R., Bougoulias, D. (1987): "Optimal Decision Fusion in Multiple Sensor Systems," *IEEE Transactions on Aerospace and Electronics Systems*, AES-23 (Sept. 1987), 644-652.

Network and Human Models for Acoustic Classification

Nelson F. Steele
ARD Corporation
9151 Rumsey Road
Columbia, MD 21045
(301)997-5600

1.0 Introduction

Despite strong effort and significant advances in automated patten recognition, people remain superior at the general task of recognizing and classifying patterns. The methods people use are the result of evolution in an environment in which aural information carries important information, and people have learned to extract that information from the acoustic waveform. How people process this data so well is not easily understood. Neural networks also show strong capability in pattern recognition and, although networks are not accurate models of neural processing, they approximate such processing better than traditional algorithmic techniques.

This effort compares human and automated pattern recognition, specifically the classification of acoustic transients into several categories by human listener and by neural network. The relative performance of human and network classifiers is evaluated first, then human classification strategies are studied, and current work compares the strategies employed by people and networks. The research described here was sponsored by the Naval Air Systems Command (NAVAIR) and the Office of Naval Research (ONR).

Specifically, these efforts have focused on the classification of different configurations of target models in an underwater environment. The contract to NAVAIR entailed investigating 1) the capability of neural networks to perform these acoustic classification tasks; 2) how well human subjects performed on the same types of tasks; and 3) the strategies employed by people as they differentiated the signals aurally. The signals used in these tasks were acoustic reflections from targets suspended in a large water-filled tank. The current contract to ONR expands on the NAVAIR research with the introduction of bottom reverberation in the signals. The first goal here is to evaluate human and network performance in the classification task for the more complex data set. The expectation is that one or both will meet with success, following which the goal is to determine what methods each of them uses in performing the classification.

2.0 First Acoustic Signal Set

Gorman and Sejnowski [1988] demonstrated that networks were capable of classifying active sonar returns into their two classes of origin. One of the goals of the studies described here was to expand the number of classes beyond two, to create a more challenging task for the

networks. Another goal was to create a more difficult acoustic condition under which the signals must be classified. Separate signal sets were created to satisfy each of these goals. The first signal set consisted of active returns from two cylindrical, enclosed, hollow, metal targets. These are four inches long and 3/4 inches wide. The two targets differ only in their shell thicknesses. The wall of one of the targets is 5% of its outside diameter, and the wall of the other target is 10% of its outside diameter (the outside diameters of each target are the same). *Shell thickness* is the first of three target parameters which combine to form the classes of signal. Each of the targets was insonified while filled with air, then water, then a solid epoxy compound; thus *Content* was the second parameter. Each target, with each content, was insonified while suspended in the water column at several different orientations with respect to the common axis of the transmitter and collector. Three of these angles were selected for experimentation. Thus *angle* was the third parameter. The combination of the three parameters produced 18 distinct classes of signal.

These targets were suspended in a laboratory tank as shown in Figure 1. The targets were insonified at several frequencies in the range of 200 to 800 kHz with six cycles of a sinusoid using a USRD Type E8 Transducer. The 200 kHz signals were selected for use in the experiments. The reflected wave from the target was picked up on a CTP LC-5 hydrophone and digitized at 2 MHz over 12 bits on a Nicolet 2090-111A Digital Oscilloscope. For each combination of all the parameters, 32 individual signals were recorded, consisting of 1024 points per signal.

500 points were extracted from each signal starting shortly before the specular, whose onset was identified reliably by a simple energy-detection algorithm. For network training each resulting signal was individually normalized to the range X(X-l, 1). An example signal is shown in Figure 2, along with a point-by-point average of the 32 signals which demonstrates the ambient noise level in the signals.

3.0 Neural Networks for Acoustic Classification

3.1 Network Training

Signals were presented to separate networks in time domain, frequency domain, and as spectrograms. Time domain input was simply the normalized digital values, with the specular returns aligned. To produce frequency domain input the signals were padded to 512 points, the FFT was applied, spectral magnitudes were computed, and the magnitudes were averaged to produce a 64 point input to the networks. To produce spectrogram input 7 overlapping windows of 128 points each were taken from each 512 point signal, their FFTs computed, spectral magnitudes taken, and the magnitudes were averaged to 16 points per window.

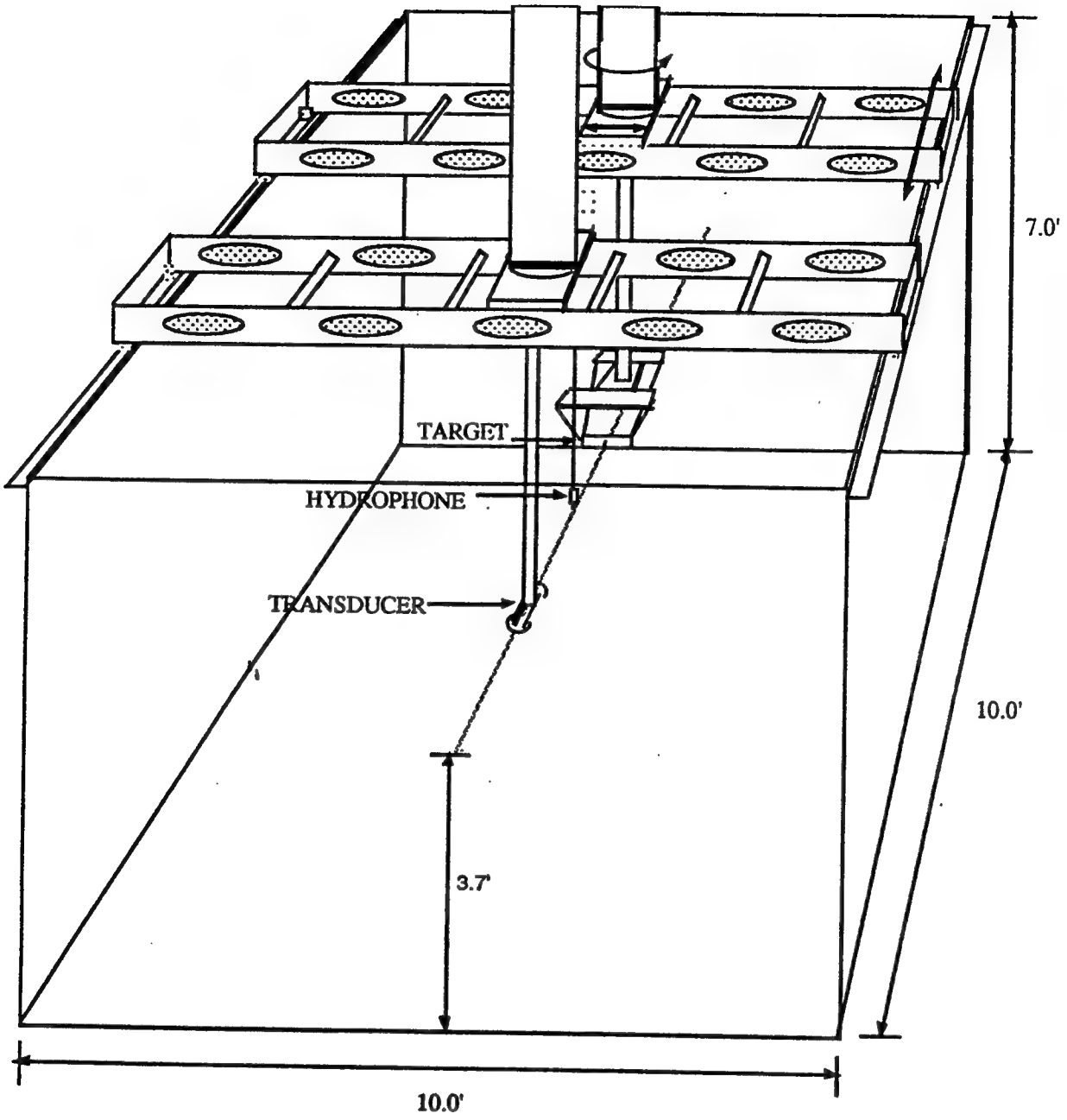


Figure 1 - Target Suspended in Tank

Standard three-layer backpropagation networks were trained to classify signals in each input form. The size of the input layer varied as described above. The hidden layer varied from 2 nodes to about 15 early in the experiments, but it was soon clear that 6 hidden nodes were sufficient to allow the network to learn to classify the signals. The first set of networks was

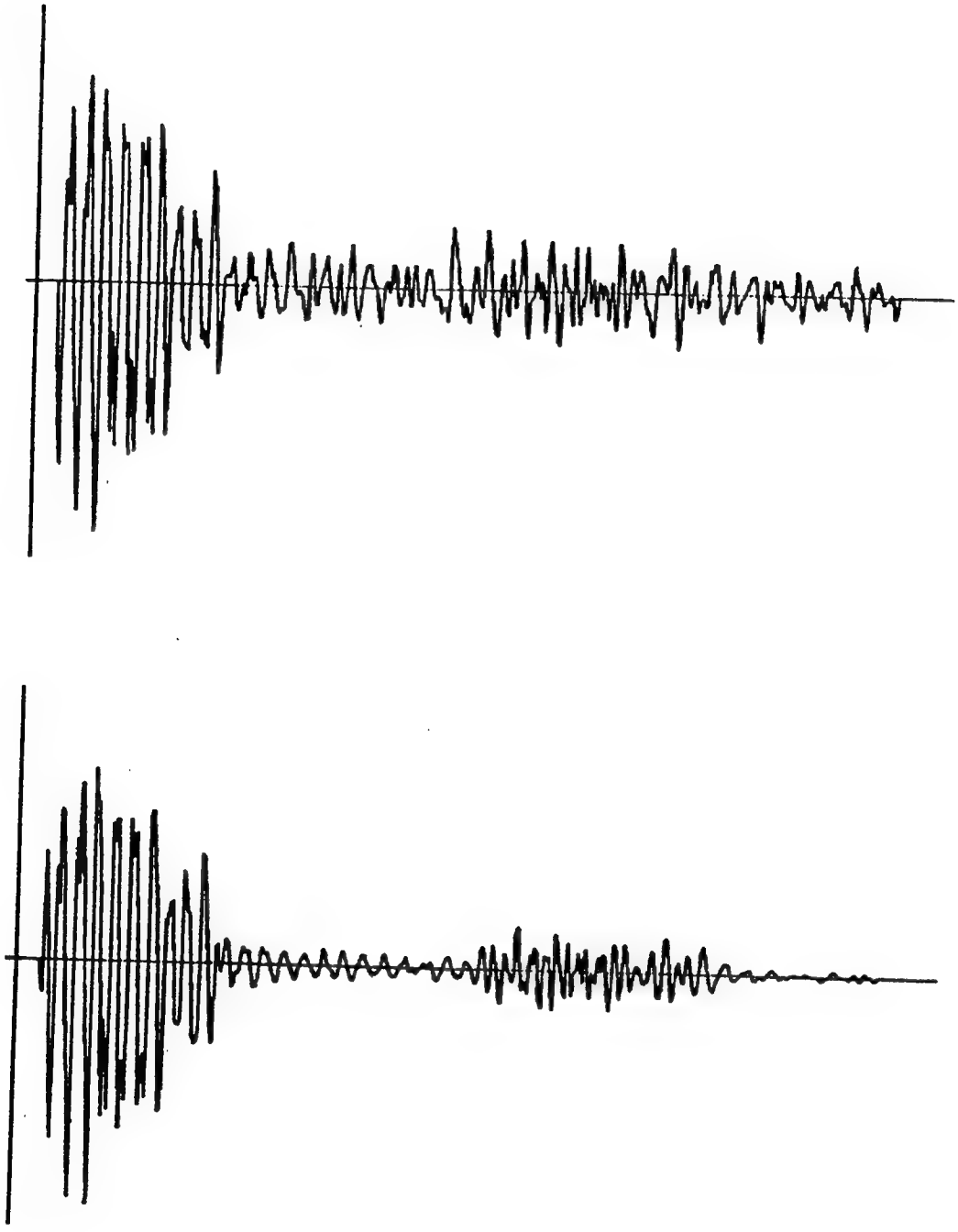


Figure 2 - One of 32 Signals (top), and the Average of the 32 signals (bottom)

trained to classify only one parameter per network. When this proved successful, a single network was trained to classify all three parameters simultaneously. In the latter case eight output nodes were used, one for each possible value of each parameter:

- Air
- Water
- Solid
- Thick Shell
- Thin Shell
- 90 °
- 45 °
- 0 °

For classification purposes an output node was trained to produce high activation when the input was a signal with the corresponding parameter true; otherwise the output node was trained to give low activation. For example, when a signal produced by insonifying the thick-shelled target is presented to the network, the desired output is high for the "thick" node and low for the "thin" node. In this manner the 18 classes are distinguished. The network training sets consisted of 16 of the 32 signals in each class. One of the remaining signals in each class was used in a validation set. Training was halted when the mean-squared error (MSE) of the validation set reached a minimum. Initial training runs led to the use of a learning rate of 0.3 and a momentum of 0.5.

Each network with at least four hidden nodes was a perfect classifier of the signals in the test set, regardless of the form of input. The only apparent differences in these networks was that those trained with frequency domain input needed consistently more iterations to reach their minimum MSE, and this MSE was higher than that for input which contained time information (including spectrograms).

3.2 Performance Under Low SNR

The 32 signals in each class exhibit a small level of ambient noise. Clearly this was not a problem either in training the networks or in their classification performance. However, the performance of a pattern recognition technique must be evaluated in light of real-world difficulties such as degraded signal-to-noise ratio (SNR). In order to measure the performance of one network under lower SNR conditions a set of time-domain signals of varying SNR was created. For each class of signal eight random sequences were created from a Normal distribution with a mean of zero and a standard deviation of 0.3. The point-by-point averaged signal in each class was computed, such as that shown in Figure 2. The averaged signal in each class was multiplied by the scaling factors 1.6, 1.4,..., 0.2, and the resulting signal was added to one of the eight random sequences. Using

$$\text{SNR} = 20 * \log (\text{scaling factor} / 0.3)$$

the resulting SNR values are

Scaling Factor	SNR (dB)
1.6	14.6
1.4	13.4
1.2	12.0
1.0	10.5
0.8	8.5
0.6	6.0
0.4	2.5
0.2	-3.5

The averaged signal was also added to this testing set. Since the noise sequences are random, a network could be expected to correctly classify one corrupted signal yet incorrectly classify another corrupted signal at the same SNR. Therefore, this process was repeated 20 times for each class, using new random sequences for every resulting signal.

One of the time-domain networks was tested against this signal set. The resulting classification performance is shown in Figure 3 for each parameter separately, and overall (all three parameters correct). Starting from perfect performance on the averaged signals, performance declines gradually and steadily as the SNR drops, with no precipitous declines. At the lowest SNR, performance is still above chance. These results against noisy signals indicate that the network is identifying signal features which are not completely obscured by substantial amounts of noise.

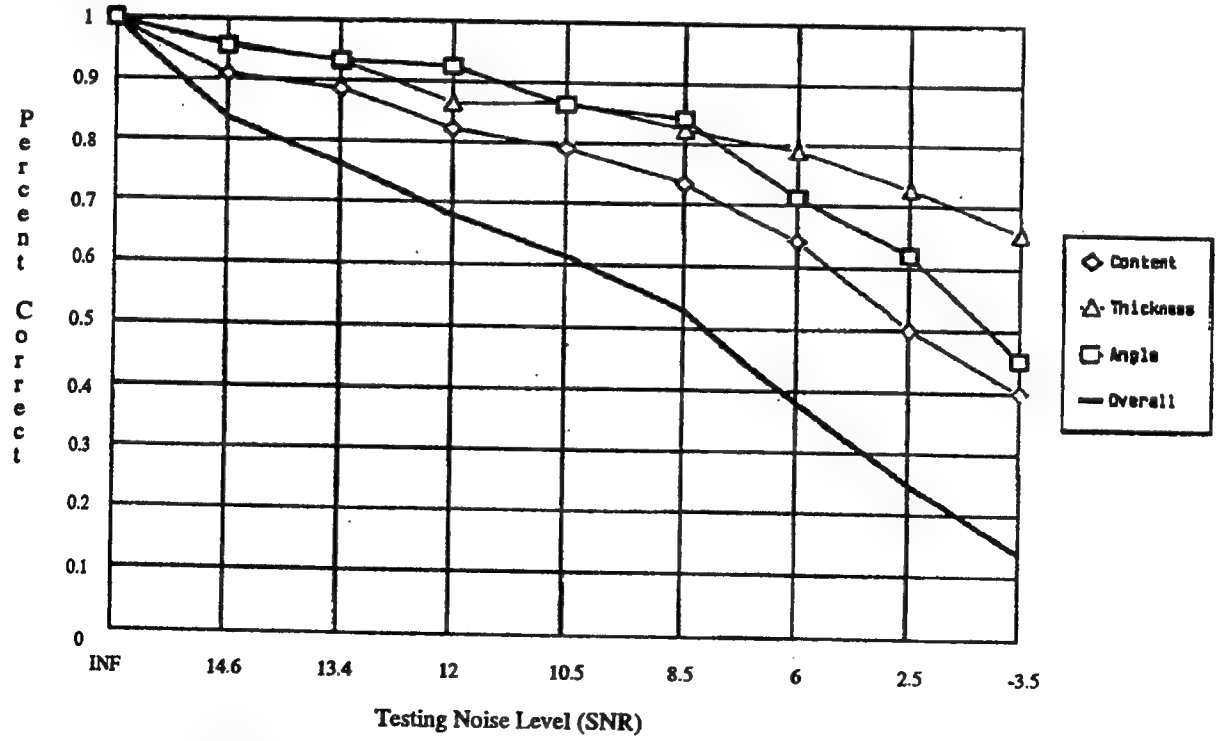


Figure 3 - Classification Performance of a Network as a Function of SNR of Test Signals

3.3 Training with Low SNR Signals

Thus far the training sets for the networks contained a rather small level of ambient noise, which may have had some effect on the classification performance of the trained networks. To measure the effect of noise in training, time-domain input networks were trained using noisy signal sets. One network was trained for each of the eight noise levels described above, by obscuring the averaged signal with random noise from the same distribution used previously. Each time a training presentation was made, a new random sequence at the given noise level was added to the averaged signal. Otherwise, training followed the regime described earlier.

As shown in Figure 4, networks trained with low SNR signals performed much better when classifying low SNR signals than networks trained with the original instances of the signals, which were of relatively high SNR. Similar results were obtained from networks trained at several noise levels, with poor results only at the lowest two noise levels. Figure 4 shows overall performance for the best two networks, trained with signals at 13.4 and 8.5 dB, as well as the overall level of the network trained with the original instances of the signals. (Figure 5 shows one signal in both its averaged form and at 8.5 dB). Classification performance on every SNR is improved, with dramatic improvements on all but the lowest SNR. A much more robust network resulted from adding substantial noise to the training set. Apparently some amount of "specialization" occurred in the network trained with high SNR signals, even though training was controlled by a validation set of unique instances. Substantial noise levels produced a more general solution.

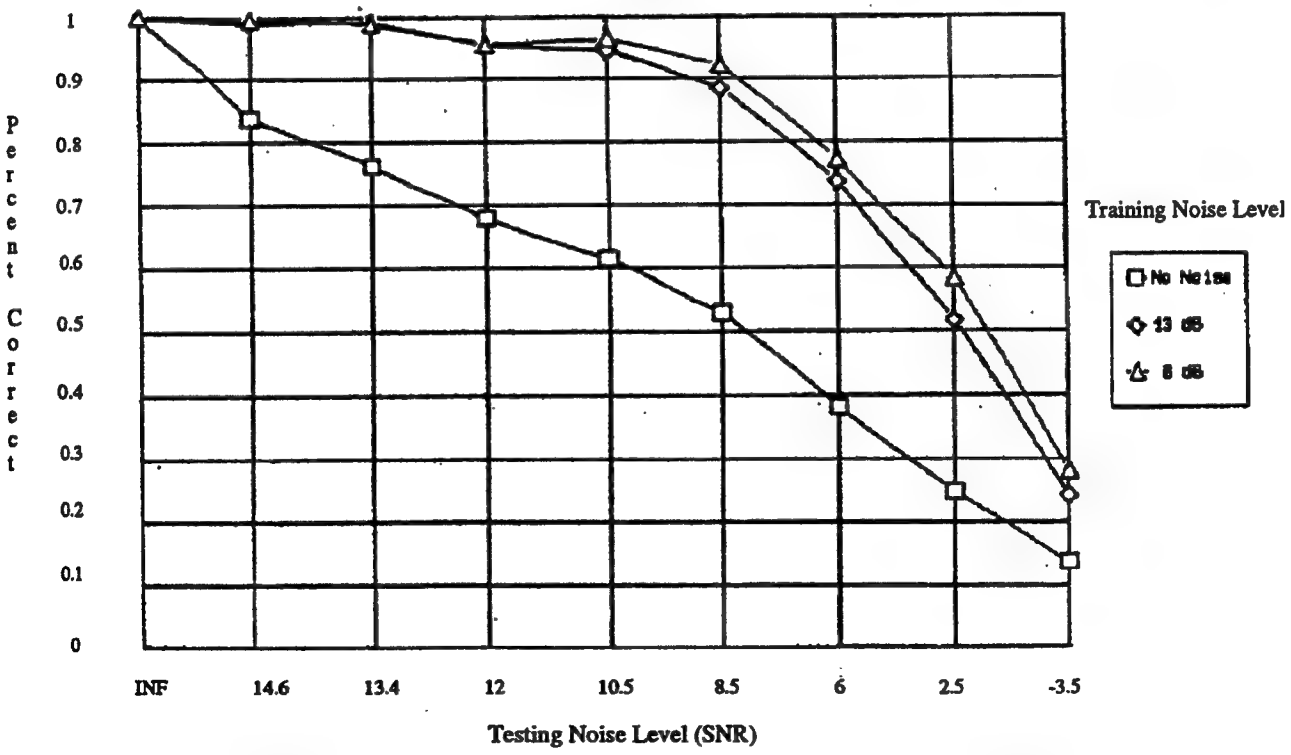


Figure 4 - Classification Performance of a Network Trained With Low SNR Signals

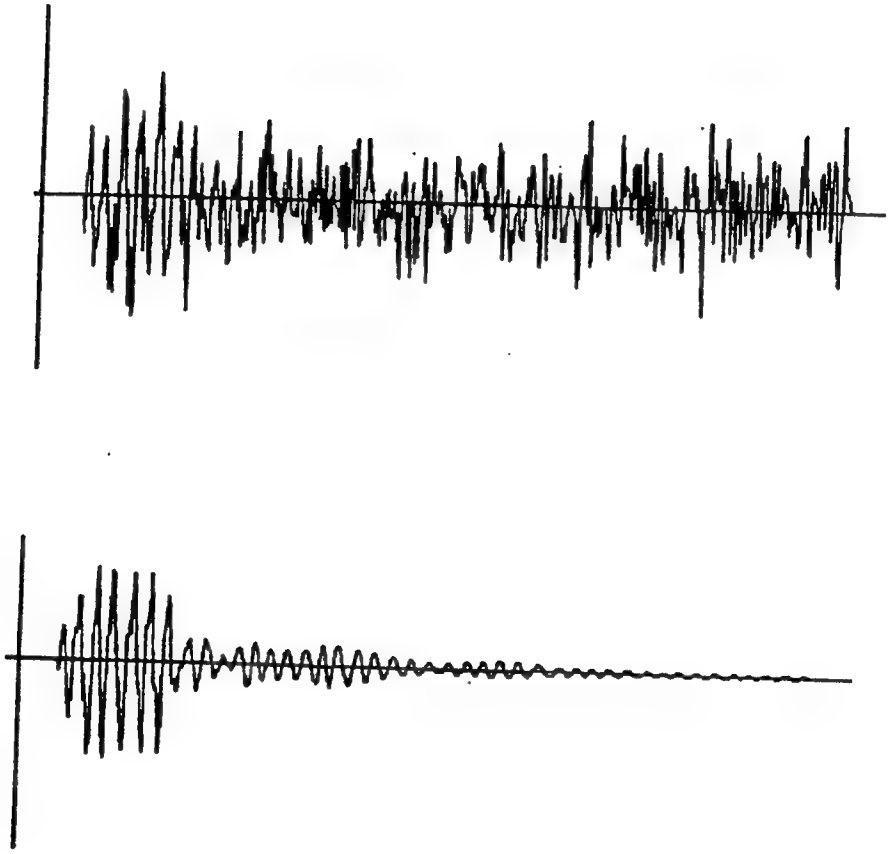


Figure 5 - A Signal in its Averaged Form and at 8.5 dB SNR

4.0 Classification Models Based on Human Listeners

In addition to the exploration of the capabilities of neural networks in discriminating the target signals, the ability of human listeners to classify these returns was investigated. The goal was first to determine that humans could perform the task, and then to understand the acoustic features in the signals that were key to this success. Multidimensional scaling was applied to the human performance data to make this feature determination. Once the scaling dimensions were established, mathematical models were created to represent them. In this way, simple models were developed of the acoustic processing used by humans in classifying the signals. Lastly, the results of the models were compared to those from the networks and evaluated for guidance for future related research.

4.1 Human Testing

Human listeners were asked to classify, and to rank the similarities of, the 18 signals described in Section 2. Both tasks involved presenting the humans with audible versions of the signals, which were created through interpolation. This was necessary to lengthen each signal to a reasonable duration and bring out distinguishable characteristics. The result was signals which had a duration of 0.4 seconds and were played at 10 kHz.

The first human experiment required the subjects to determine the three target parameters of content, thickness, and angle. As was the case with the networks, the task was successfully accomplished by several of the human subjects tested. That humans could distinguish signals of high SNR laid the foundation for the next experiment and the exploration of its results. In the second experiment the listeners judged the relative similarity of the signals. The purpose of this experiment was to provide data which would be used to analyze the strategies employed by the subjects in identifying the signals.

4.2 Scaling

The means of estimating the strategies used by the subjects was to derive patterns or structures which were acoustically defined in the signals. The method of applying a multidimensional scaling (MDS) algorithm to the similarity data was used as a first step toward this end. MDS provided a means of scaling the similarity ratings to a manageable number of dimensions in a "feature" space. In this case, four dimensions were needed to account for sufficient variance in the similarity data. The result was a four-space point for each of the 18 signals, which defined the signal's relative position in so-called feature space. Once the dimensions were finalized they had to be interpreted, with respect to the signals' acoustic characteristics, in order for the models of the features to be created.

4.3 Modeling Human Strategies

The models of the scaling dimensions were determined in different ways. Observation of the

signals' characteristics, both visual and auditory, provided information that led to the first and fourth dimension models. The second and third dimension models proved more difficult to resolve so a unique approach with neural networks was used in finding solutions. Networks were trained on the signal set such that the target output for each signal was its position along the dimension in question. The weight structures of each network, whose output had a very high correlation to a single dimension, were then evaluated to derive the means by which the network mapped the acoustic information to the dimension value.

The evaluations of the scaling results revealed that all input forms used in the classification networks were also involved in the humans' decision making processes. The dimensions, ranked from first to fourth, are in order of their role in accounting for the variance in the scaling solution. The first dimension was found to be a center of gravity point along the time-domain's x-axis. The second and third dimensions involved both time- and frequency-domain information. Both dimensions depended on a small set of frequency bins at specific time intervals over the duration of the signals. Finally, the fourth dimension relied only on frequency-domain information, and was a simple ratio of low to middle range frequencies in the signals.

These four models were then used in conjunction to determine the relative placement of an unknown signal in four-dimension space. Thus, the models geometrically provided the means to classify signals which were not included in the original set. In other words, given a 4-space point composed of the values of all four models for an incoming signal, the Euclidean distance from this point to each of the 18 4-space points representing the known signals could be computed, and the resulting distances could be used to classify the unknown signal.

The models performed well above chance, although they were not perfect classifiers. There were particular problems with the cases where there was a cluster of signals which the humans had had difficulty distinguishing. This difficulty appeared in all dimensions, and thus in all of the models. Under low SNR conditions, classification using these models was poor. In general, though, the modeling of the human processing of the signals was quite successful as exploratory work.

4.4 Human Model vs Network Performance

The success of the human models was compared to that of the networks that were discussed earlier. The models did perform well on clean signals, but did not do well with signals which were embedded in a noisy background. Network performance exceeded that of the models, particularly under noisy conditions. The differences in performance can be attributed to several factors, including the limitation of the models in representing human classification strategies, and the relative robustness of a network that has been successfully trained. Further study of modeling methods, ways of breaking down the signal set for problem clusters, and different classification rules would most likely increase the performance and usefulness of this approach to signal classification.

5.0 Network Strategies Under Reverberation Conditions

Having quantified the performance of networks in classifying this signal set and investigated human classification strategies, current efforts add two major goals:

- Derive the strategies used by successful networks to classify the signals, and compare these to human strategies.
- Use a more complex signal set to further challenge network classification capabilities.

The latter goal was achieved by collecting signals from targets which, in addition to being suspended in the water column, were set on a smooth sandy bottom (as seen in Figure 7). The resulting reverberation is a significant complication in classifying the signals. An example of such a signal is shown in Figure 6.

Because the reflected wave from the bottom obscures the return from the signal, there is no obvious specular reflection with which to align the signals in time. Time domain alignment is a laboratory adjustment in any event, and it is preferable to avoid it. Another problem with the time domain representation (without some further processing such as enveloping) is the number of free parameters available to the network. The use of one arc and one weight per input point, given the large number of input points, may give the network enough parameters to learn arbitrary classification strategies unrelated to the actual class of the signal. The network may only memorize" the input set while capturing no information about the signal classes in general. While adding noise to the signals may alleviate this tendency, a more fundamental solution is sought.

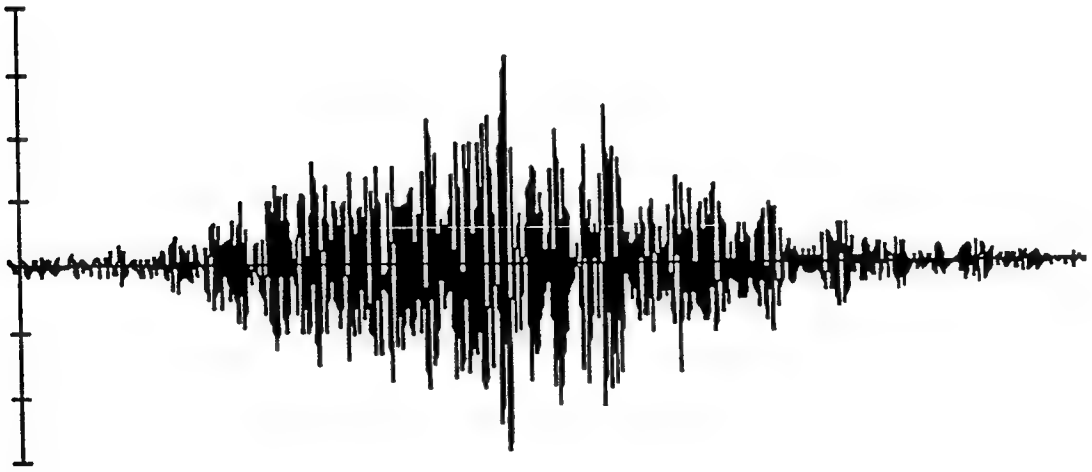


Figure 6 - Signal from Target on Sandy Bottom

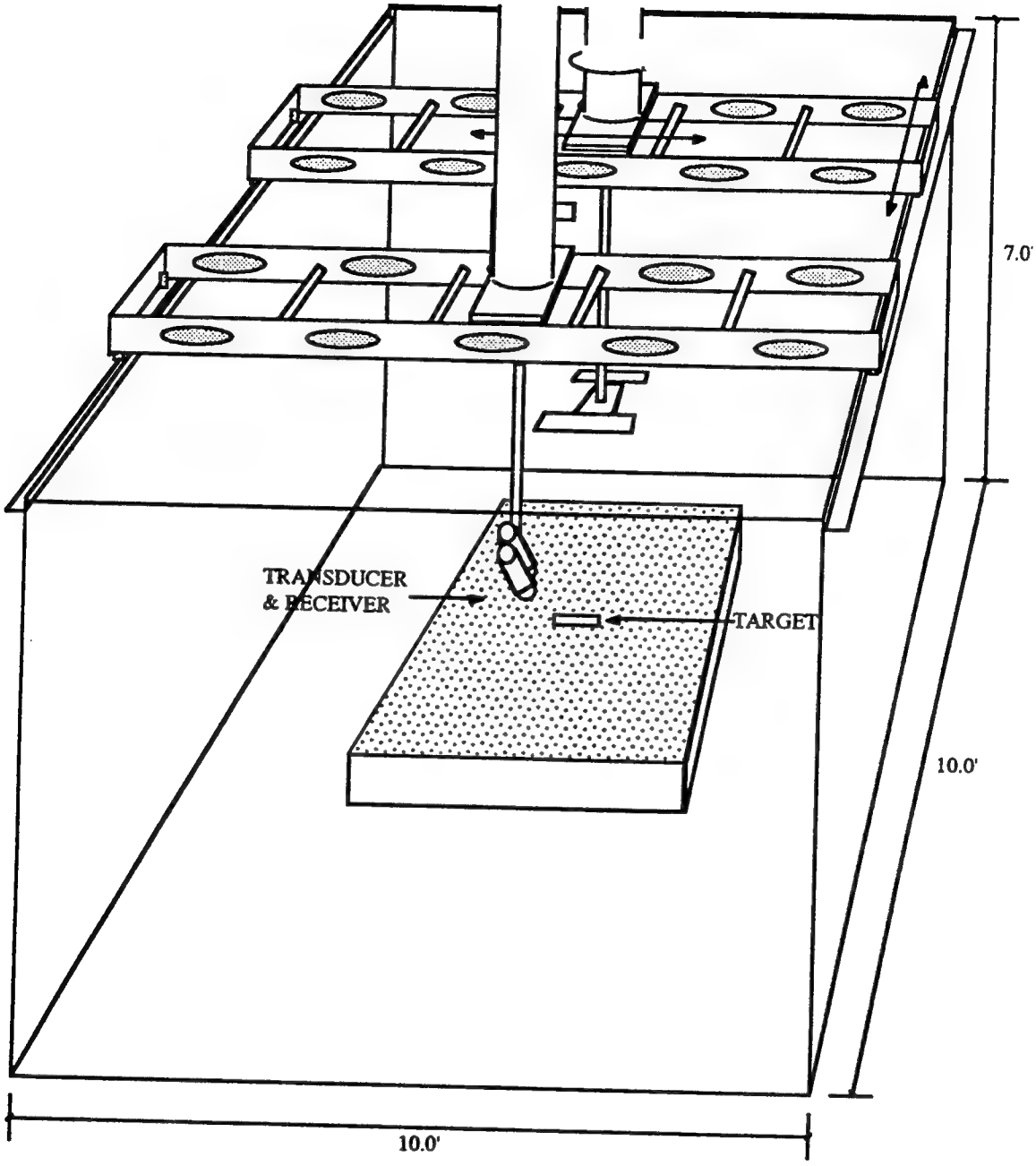


Figure 7 - Target on Sandy Bottom in Tank

5. 1 Recurrent Networks

One such solution may be found in recurrent networks. Under this scheme a signal is presented to the network one point at a time, using only one input node. The network feeds intermediate results back to the hidden layer, in the manner of Elman [1988]. The storage of intermediate results may be extended back beyond the previous cycle, which is likely to be necessary because the signal develops over time and signal features can be expected to occupy many digitized points.

In the first step towards this goal the Exclusive Or (XOr) problem is configured for a recurrent network. The input is a random binary sequence which is applied to a single input node. The network acts as a normal backpropagation network, with the addition of a recurrent layer whose size is equal to the hidden layer (see Figure 8). The hidden nodes send their output to the recurrent layer over arcs with fixed weights of 1.0. The recurrent layer is fully interconnected back to the hidden layer. The result from the single output node is the XOr function computed from the current input and the previous input.

A recurrent network with four hidden nodes (and therefore four recurrent nodes) was trained to perfectly compute the "recurrent XOr" function. Very small learning rates, on the order of 0.01, are needed.

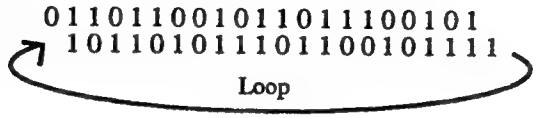
5.2 Sequential Input Shift

Another means of avoiding the problems of time-domain alignment and of having too many parameters is to use a standard network but shift the signal across the input nodes, one node per cycle of the network. This is illustrated in Figure 9. The size of the input layer need only be as large as the feature needed to identify the signal. The drawback is that one may not know a priori how many points such features may occupy. Initial results show that 30 input nodes are sufficient to classify some of these signals. The next effort will be to extract the classification strategy of such a network.

INPUT: Random Binary Sequence X_i

OUTPUT: XOr (X_i, X_{i-1})

Example:



NETWORK:

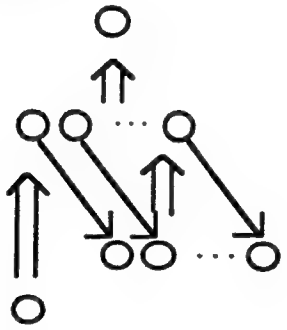


Figure 8 - Structure of the Recurrent Network

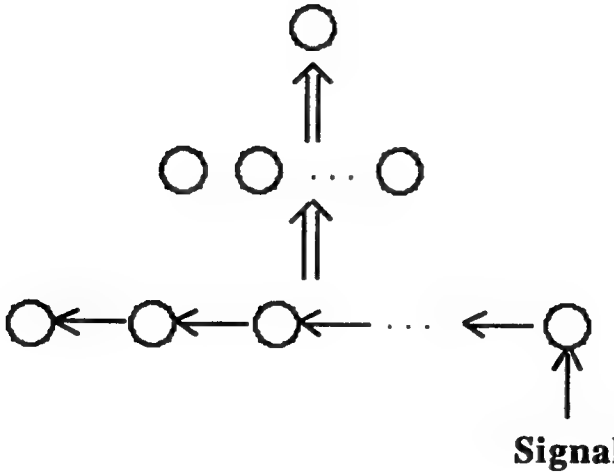


Figure 9 - Sturcture of the Sequential Input Shift Network

6.0 Conclusion

Neural networks were sometimes compared to brains in their non-sequential means of processing information. While the wilder claims were exaggerated, it is still interesting to compare human strategies in the classification task to the strategies derived through network training. Since coding network input is often critical to a network's success, knowledge of what signal transforms are done by the human listener to distinguish the sounds may be useful in guiding network development. Conversely, networks show higher performance under severe noise than human listeners. How the networks accomplish this is of interest not only in its own right, but also in comparison to human processing under similar circumstances.

Neural Network Automatic Gain Control for Low-Cost IR Sensors

Andrew J. Moore, Massimo A. Sivilotti, John Tanner
Tanner Research
180 N. Vinedo Ave.
Pasadena, CA 91107(818) 792-3000

1.0 Introduction

Infrared sensors with excellent dynamic range have been developed for government applications. Typical state-of-the-art sensors can respond to signals over a 3 order of magnitude range. Video monitors, on the other hand, can only display intensities over a range of about 1.5 orders of magnitude. If the sensor signal is scaled down to match the dynamic range of the video monitor, the displayed image has very poor contrast.

This problem is especially obvious for scenes in which one area is much brighter than the rest of the scene. Flames, for example, will overload the monitor range unless the sensor is "turned down" by reducing the iris aperture or by shortening the pixel integration time. With the sensor turned down, though, it is difficult to see more than a silhouette of what is burning; important details, such as the numbers on a vehicle, for example, are lost in darkness.

As another example, consider the result of imaging the desert at night from a helicopter. The desert floor is still "bright" in the IR band, since it is still warm from the heat of the day, but the night sky is cool and dark. It is difficult to image detail in these two disparate regions. Power lines in the flight path may not be visible -- with tragic consequences.

2.0 Spatial Filtering to Compress Dynamic Range, a Natural Solution

Photographic scientists have faced a similar problem for many years. The dynamic range of photographic negative materials is about 3 orders of magnitude, but the range of print paper is about 1.5 orders of magnitude. The conventional solution to this range mismatch is high-pass filtering. The image is blurred (low-pass filtered) and subtracted from the original. This has the effect of "pulling" bright regions and "pushing" the dark regions into the middle of the paper range. Regions that were out of range are brought into range with this technique.

Some may argue that the eye and brain carry out similar processing. We see small overshoots that presumably result from high-pass filtering around small steps in intensity -- these overshoots are called *Mach bands* (Ratliff, 1965). As such, spatial high-pass filtering may be viewed as neural network processing. This is not the whole story, however. If the eye and brain perform linear high-pass spatial filtering, we should also see large overshoots at large steps in intensity. For example, as we look from a dim office out a window at a sunlit scene, we should see a bright overshoot above the windowsill and a dark overshoot below it. But we do not (Ratliff, 1985).

3.0 Nonlinear Spatial Filtering

A neural network that faithfully models the response of the eye, then, should filter linearly at small steps in intensity, but not at large steps in intensity. At Tanner Research, we have designed, fabricated, and tested a neural network image processing integrated circuit that operates in this way. As demonstrated at the presentation (Navy SBIR Neural Network Conference, Arlington, Virginia, June 4, 1992), overshoots at a large step in intensity (such as a windowsill) are not produced in an imaging system that uses this processor.

The prototype VLSI processor performs real-time *nonlinear* spatial filtering at the video rate. The chip uses standard CMOS technology, and consequently represents an inexpensive component of an IR imaging system. The present chip has a 50 by 50 pixel resolution. The image is subsampled for input to the chip, and the chip output is then subtracted from the original image. Note that after subtraction, the resulting image has the high resolution of the original image.

4.0 Automatic Gain Control

As part of our Phase 1 SBIR contract (N60530-91-C-0260) with the China Lake Naval Air Warfare Center, Weapons Division (Howard McCauley, technical monitor), we are building a circuit board that accepts a standard RS-170 video signal and interfaces it to our neural network chip. Several automatic gain control techniques are implemented on the board, so that each may be evaluated. We have demonstrated the usefulness of the processor with both CCD and PtSi IR sensors. (We are grateful to Dr. Hammam Elabd and his colleagues at Loral Fairchild Imaging Sensors for assistance in obtaining the IR imagery.) For both types of input we have shown that the system performs effective range compression and background removal. Also, we have shown that the system can reduce the halo around a hot point source that is produced by PtSi sensors. This is of crucial importance in detecting the "hard body in a plume."

5.0 Future Research

We intend to continue development of the image processor and associated gain control circuitry in the future. In particular, we plan to increase the chip resolution to VHS resolution (250x250), to integrate automatic gain control circuitry, and to reduce switching noise. We also intend to engineer a tightly integrated interface between the neural network processor and a particular IR sensor.

6.0 References

Ratilff, F., *Mach Bands: Quantitative Studies on Neural Networks in the Retina*, San Francisco: Holden-Day, 1965.

Ratliff, F., "Why Mach bands are not seen at the edges of a step," *Vision Res.* **24**: 163-165, 1984.

Automatic Detection and Classification of Marine Mammal Underwater Acoustic Emissions using Neural Networks

Jerry L. Fuqua, Ph.D. and Reid H. Shizumura
ORINCON-Hawaii Corporation
970 North Kalaheo Avenue, Suite C-215
Kailua, Hawaii 96734

Abstract: A study was conducted with the objective of detecting and classifying underwater acoustical emissions from marine mammals using a feed forward back propagation artificial neural network. The work described in this paper focuses on the application of signal processing and neural network techniques to automatically detect, classify, and discriminate these sources of acoustical energy from other natural, or artificial sources. A three stage signal processing and neural network classification system was applied to several hours of ocean acoustical data to extract exemplars for training and for subsequent classification testing. The results indicate that the developed back propagation network and the associated weight set produced an effective mechanism for successfully detecting and classifying mammal acoustic emission energy from the acoustical background.

1.0 Introduction

The ambient acoustical background of the ocean contains a number of biological sources which are readily detected by hydrophone arrays. Presently it is felt that only three groups of marine animals make significant contributions to the background noise level: certain species of fish, shellfish (Crustacea), and marine mammals (Cetacea) [1]. The marine mammal group consists of dolphins, porpoises, and whales, of which the latter are believed to produce highly energetic, long range emissions. These emissions contain a series of low frequency pulses which can last from minutes to several hours in duration. The individual pulses are on the order of a minute in length and can vary in frequency with time (dynamic) or be constant in time (non-dynamic).

For several years the discrete dynamic events were believed to be whale produced. Often referred to as "commas" due to their characteristic similarity to the punctuation symbol on acoustical spectrograms, these emissions are often found to be in competition with other emissions within the same range of frequencies. The resulting spectrogram may be viewed as a representative energy environment consisting of biologically produced feature information among acoustical clutter. From the perspective of the defense community these emissions are viewed as components of the acoustical clutter itself. It is desirable to identify and remove these features in order to improve upon the detection and analysis of the signals of interest. To help extract out this biological information an appropriate data processing and analysis system is desired.

Since these characteristic features are found in the spectral domain, somewhat traditional signal processing and conditioning methods are appropriate for the initial manipulation of any ocean acoustical data. Subsequent detection and classification of commas may be viewed as a binary discrimination from non-comma, acoustical clutter. Since traditional classification and discrimination techniques often produce poor results with such data, the situation seems appropriate for artificial neural network (ANN) application. In particular, since the problem is one of a complex input with binary classification, a feed forward back propagation paradigm would be suitable.

2.0 Methods

The complete processing procedure required for feature extraction and classification consists of the three functional stages illustrated in Figure 2.1. These stages involve signal processing, signal conditioning, and detection processing. In the first stage the acoustical time series is transformed into the frequency domain by a Fast Fourier Transform (FFT). The resulting spectral data goes through frequency and temporal normalization during the signal conditioning stage. At this point the spectral clutter has been minimized, enhancing the features for detection and classification. During the detection stage the conditioned input is presented to the neural network through a data retina. The generated neuron excitation levels serve as the basis for detection alerts and classification.

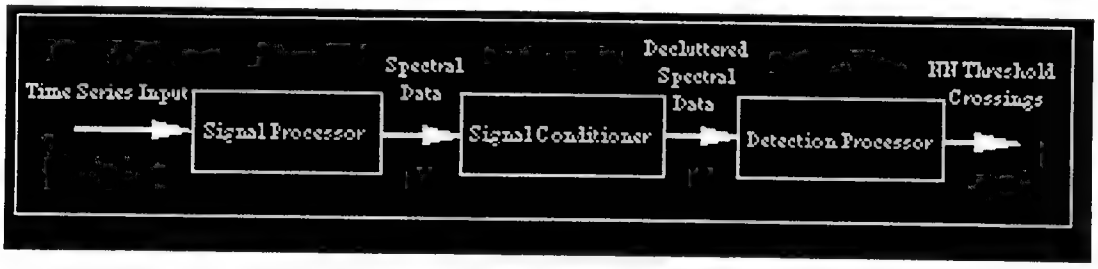


Figure 2.1: Functional Signal and Information Flow Diagram

A detailed breakdown of the three processing stages is presented in Figure 2.2. During signal processing the original time series is partitioned into 1024-sample blocks, representing 5 second scans. The data is then overlapped by 50% to create 2048-sample blocks for FFT processing. At this point the buffer represents a 10 second scan input for a 2048-point transform. The transformation is conducted with a sliding FFT in one block (1024 point) increments. Within the FFT the time stream is passed through a Hamming window to remove the effects of overlapping the input data. The spectral output is in the form of magnitude squared power replicas and is passed through a square root function to produce magnitude bounded data.

During the signal conditioning stage the scan level data is averaged by two frequency bins and two successive scans, producing scans at 10 second intervals. This reduces the number of output scans, and the number of points within a scan, by a factor of two. Spectral normalization of this data is performed by the noise spectrum estimation (NSE) procedure. The NSE algorithm

reduces the noise background to enhance the detection of narrow band energy. The resulting 10 second scans are passed to the temporal normalization procedure where stable narrow band signals are effectively removed. Removal of these components insures that only dynamic signal events remain in the spectral level representations. A comparative example of the effects of this processing is presented in Figure 2.3. (An informative review of relevant spectral processing and conditioning techniques is presented by Jackson [2].)

Detection is performed by a feed forward Back Propagation (BP) Artificial Neural Network (ANN), consisting of two hidden layers, each containing 10 nodes, and two output classes representing dynamic events (commas) and clutter (non-commas). This paradigm follows the somewhat standard design describe by McClelland and Rumelhart [3]. The network input retina is positioned to start with the minimum targeted frequency bin and is composed of 40 total bins and 16 scans. This retina specification, which results in 640 total inputs to the network, was designed to capture multiple comma events in the frequency domain. During processing the retina is shifted one scan at a time and resubmitted to the network. All output excitation levels are passed through an exponential filter where .85 of the preceding excitation is added to .15 of the current excitation to produce a smoothed ANN excitation stream. The smoothed excitations are then passed through a threshold function for detection alert generation.

In preparation for training the neural network, continuous spectrograms of acoustical ocean data were reviewed to select comma exemplars and noise exemplars. For comma exemplars the data was analyzed to determine the dynamic start and stop times, event duration, and the change in frequency for the event. Exemplar selection was accomplished by placing the start of each comma event in the middle of the retina, and stepping the retina down the spectrogram record in one scan increments until the entire dynamic event was captured in the training file. Noise exemplars were extracted from spectrogram records containing training commas and from records free of commas. These exemplars ranged from natural biological interference and random fluctuations, to artifacts produced from data processing.

Training the network to obtain a dynamic weight set requires an iterative procedure. The first step of the process is training the network on a limited set of dynamic event and noise exemplars until the root mean squared (RMS) error of the weight set is less than .02. Next the weight set is tested_against the time series data containing the first training events where the performance against this data is determined. In the event that the performance against the training time series is less than desired, additional exemplars are extracted and trained. Only after adequate performance is obtained against the data containing the first dynamic events are exemplars for additional events added to the training file. This process is repeated until adequate performance against all training events and ocean background noise is obtained.

The initial training set consisted of 209 comma and noise exemplars. This initial training session needed approximately a thousand cycles before the RMS error was less than .02. The iterative procedure stated above was followed until adequate performance was verified against all training data. The final weight set was obtained from a training set of 639 exemplars, of which approximately one third were commas..

Performance testing of the network consisted of processing over 35 hours of ocean data

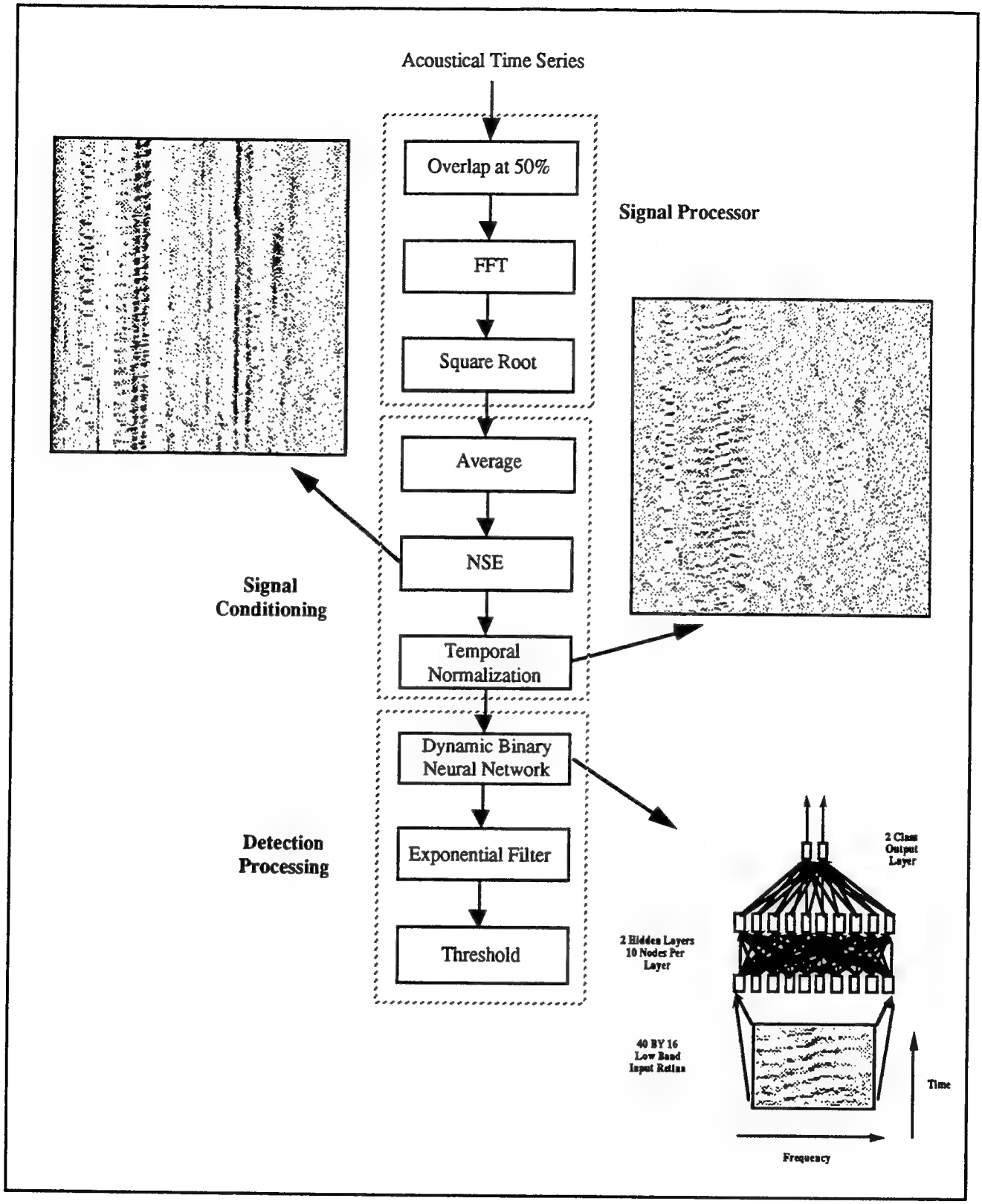
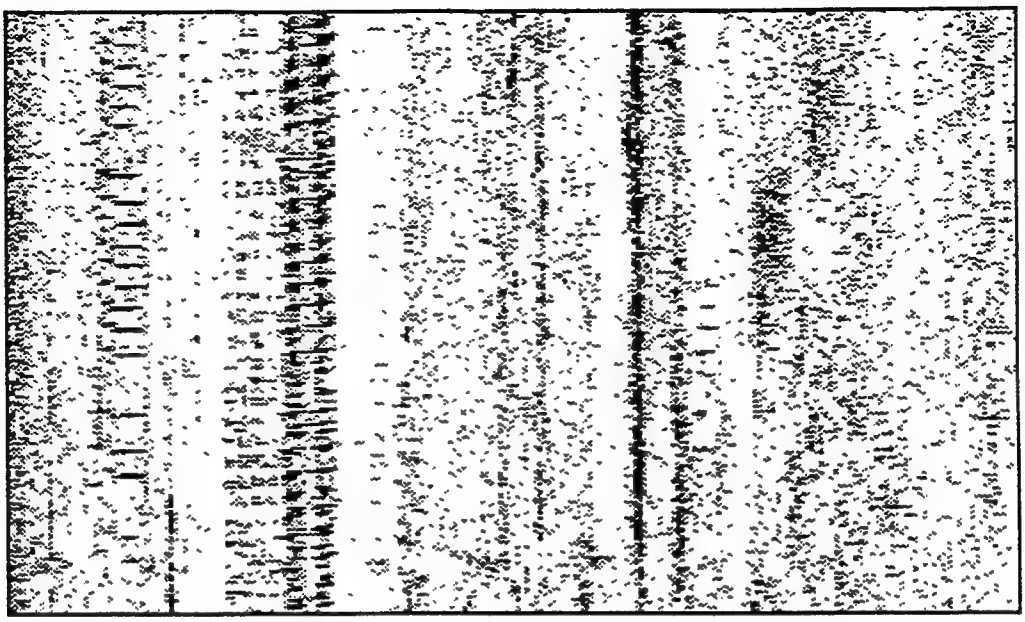
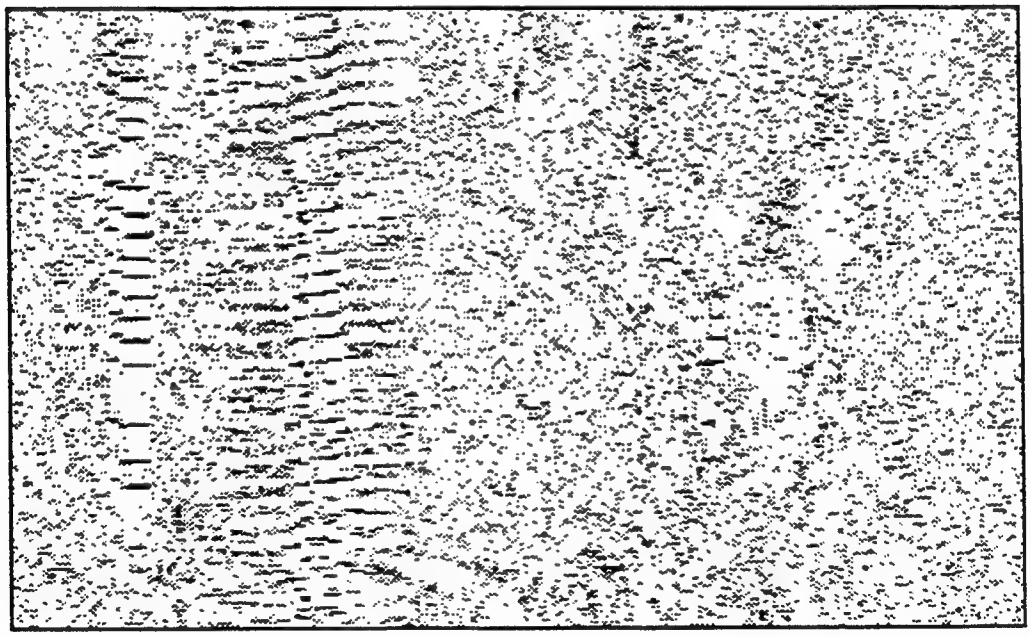


Figure 2.2: Dynamic Comma Event Detection and Binary Classification Processing Chain



Normalized Spectral Estimated



Temporal Normalized

Figure 2.3: Comparison of Normalized Spectral Estimation (NSE) and Temporal Normalization (alpha = .06) of acoustical spectrograms.

containing 148 discrete comma events. The data consisted of several contributions containing moderate to heavy backgrounds from both the Atlantic and Pacific Oceans. During the signal conditioning stage a time normalization constant (alpha) of .06 was used. From previous work this time constant was found to be successful in promoting good detection levels at reduced false alarm rates. Exponentially filtered neural network excitation levels for the comma class were extracted for each scan output and plotted against time. A typical example of such a plot is represented by the segment illustrated in Figure 2.4. In this example ground truth analysis revealed twelve comma events, of which two were found to be relatively weak. One false alarm occurred at a threshold value of 0.1.

A review of the ground truth developed for each comma event was conducted for the entire data set to locate precise times for each dynamic event. The number of events detected was manually extracted and recorded. Excitation levels from ocean noise environments were counted and recorded in a similar manner with the exception that no dynamic events of interest were located within the noise.

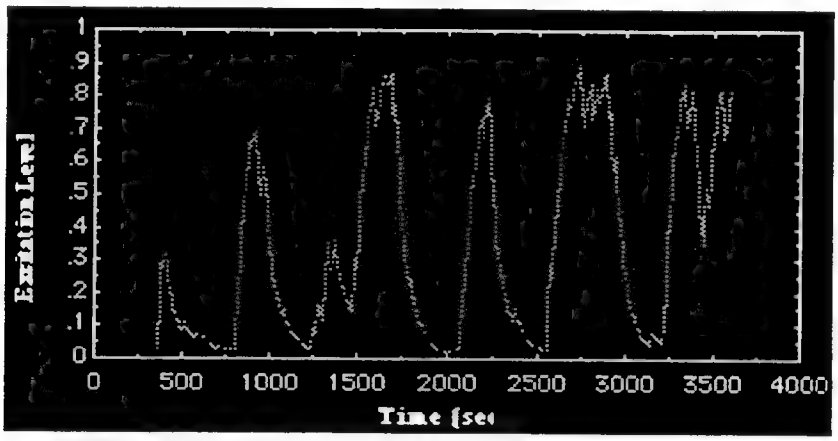


Figure 2.4: Example of Exponentially Filtered Neural Network Excitation Level vs. Time Plot

3.0 Results

Dynamic comma event detections and false alarms as a function of neural network threshold level for approximately 36 hours of data are summarized in Table 3.1. Compared to ground truth, only at a threshold of 0.1 were all comma events detected. At this level the corresponding number of false alarms was 266. With thresholds of 0.2 and 0.3 the number of detections dropped to 146 and 142 respectively. Though the drop in detections is modest, the corresponding drop in false alarms is significant, at 70 and 16 respectively.

Threshold	Comma Events Detected	False Alarms
0.1	148	266
0.2	146	70
0.3	142	16
0.4	136	3
0.5	128	3
0.6	110	2
0.7	97	0
0.8	78	0
0.9	53	0

Table 3.1: Detection Results for Comma Detection Neural Network (Alpha .06)

From the tabulated data, probability of detection (Pd) and probability of false alarm (Pfa) values were computed, and are listed as a function of threshold value in Table 3.2. The corresponding Pd and Pfa curves are presented in figure 3.1 and 3.2. These data indicate that the probability of comma event detection is quite good up to a threshold value of 0.5, and reasonable up to 0.6. The false alarm rate becomes significant at a threshold of 0.3 and below. Above this value rates were essentially negligible.

Threshold	Comma Event Pd	False Alarm Rate (hr -1)
0.1	1.0	7.44
0.2	0.99	1.96
0.3	0.96	0.45
0.4	0.92	0.08
0.5	0.86	0.08
0.6	0.74	0.06
0.7	0.66	0.00
0.8	0.53	0.00
0.9	0.36	0.00

Table 3.2: Probability of Detection (Pd) and False Alarm Rate (Pfa), as a Function of Threshold for Comma Detection Neural Network (Alpha .06)

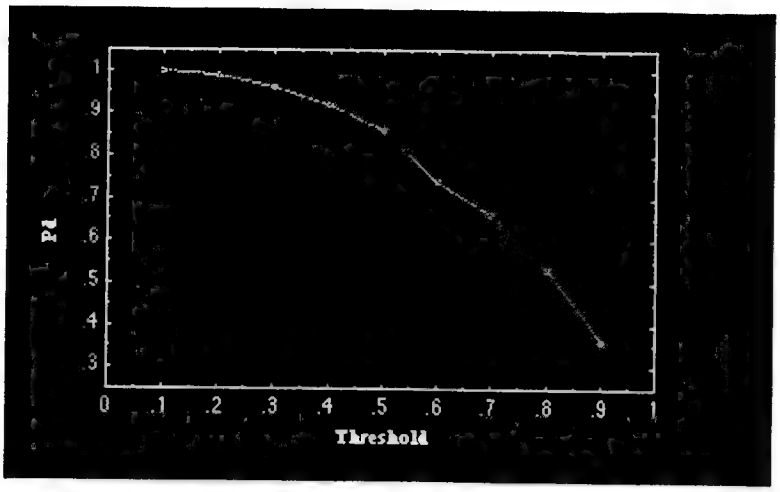


Figure 3.1: Probability of Detection for Comma Events as a Function of Neural Network Threshold

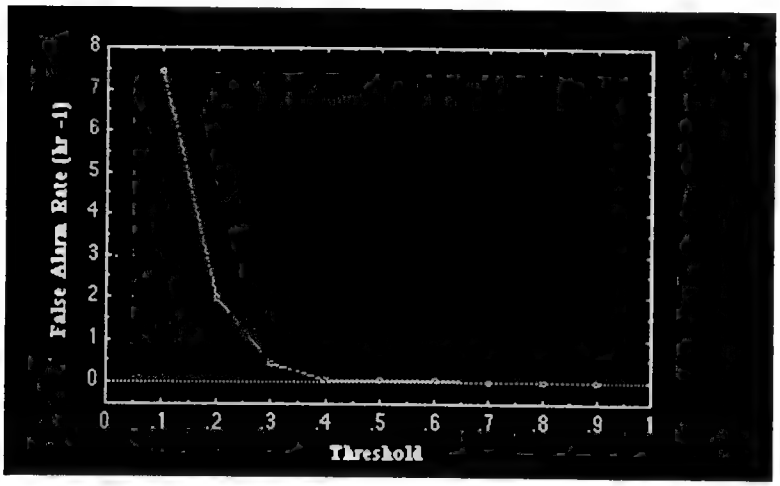


Figure 3.2: False Alarm Rate per Hour for Comma Events as a Function of Neural Network Threshold

A subsequent plot of the probability of detection versus false alarm rate is presented in Figure 3.3. This curve indicates that very good detection performance is obtained without significant false alarm rates up to Pd values about .95. Only above this value is there a significant trade-off with false alarms.

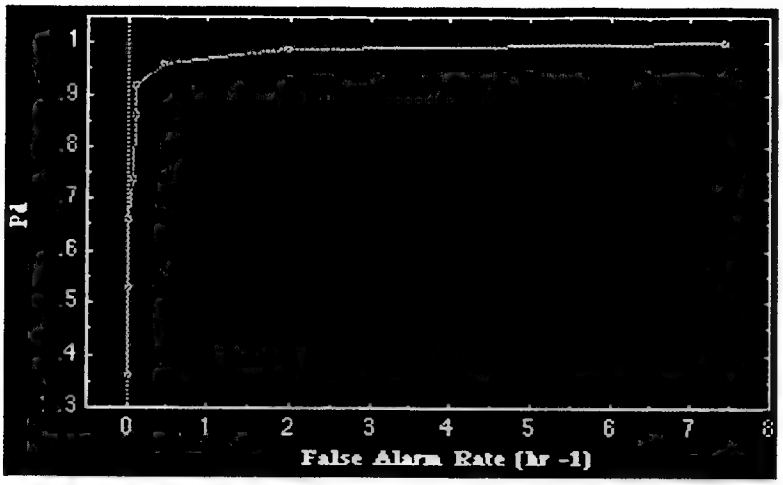


Figure 3.3: Probability of Detection for Comma Events vs False Alarm Rate (hr -1)

4.0 Discussion

The task of extracting acoustical features, descriptive of a characteristic emission, from a noisy marine environment is quite difficult in general. Without the assistance of some pre-conditioning of the data, a neural network would most probably fail. This is analogous to the human situation where an observer can detect critical features only after significant processing and conditioning has been performed. No matter how robust a network is, it would be presumptuous to expect high detection performance from raw data containing the complicating elements of noise and clutter.

In the situation presented, data conditioning was used to take advantage of the spectral nature of the features being studied. In this case temporal normalization was utilized to enhance the dynamic characteristics of comma events, by removing or reducing the non-dynamic features of the spectra. This step in data conditioning produces a much more suitable input for network training. Previous efforts with spectral normalized, but not temporal normalized, acoustical data have resulted in poor network training and subsequent dynamic detection performance.

The results of testing the developed network on over 35 hours of ocean acoustical data indicated that good to excellent comma detection levels could be obtained with low false alarm rates. Levels exceeding 90% were easily attained with false alarm rates below one per hour. A review of the Pd vs threshold and Pfa vs threshold results indicate that a value of 0.4 for the decision threshold would produce a detection level in excess of 90% while holding false alarms to less than one every 12 hours. (At a threshold of 0.3 the detection level would exceed 95%, with less than one false alarm every two hours.)

The fact that a rather simple back propagation neural network can be applied to data from a complex acoustical environment and successfully detect and classify mammal emissions can be attributed to a number of factors. Sufficient data existed to extract a large number of exemplars

which could support an iterative training process. Once the network was successfully trained on a nominal set of exemplars, successive iterative training sessions could be used to enhance network reliability and robustness. Temporal normalization of the spectral data produces the desired effect of enhancing feature definition, which in turn facilitates both training and detection. Part of the success of the network can also be attributed to the design of the input retina. The retina specification allowed for sufficient spectral information to be captured into the input exemplars while limiting the size of the input. This resulted in a smaller input layer, and subsequently a simpler and easier network to train.

The developed three-stage detection and classification system takes advantage of the strengths of traditional signal processing and spectral data conditioning, and the adaptability of artificial neural networks. This system demonstrates the potential for networks to be utilized for detecting, classifying, and studying natural acoustical phenomena.

5.0 References

[1] Urick, Robert J., Principles of Underwater Sound, McGraw-Hill, New York, 1983.

[2] Jackson, Leland B., Digital Filters and Signal Processing, Kluwer Academic Publishers, Boston, 1986.

[3] McClelland, James L. and David E. Rumelhart, Parallel Distributed Processing v. 1 and v. 2, The MIT Press, Cambridge, 1987.

Target Recognition, Tracking and Response with Neural Network Kalman-Bucy Filters

Robert L. Dawes
Martingale Research Corporation
100 Allentown Parkway, Suite 211
Allen, TX 75002
(214) 422-4570

Abstract: This paper describes the nature of the state estimation (tracking) problem for multiple simultaneous systems, and it describes our unique reformulation of the problem that admits the use of the nonlinear time-dependent Schroedinger equation in a predictor-error-corrector loop to compute the time varying probability density function for the state of the observed system(s). We show the results of simulations that demonstrate (1) the ability of the algorithm to automatically detect multiple targets even at 0 dB SNR; (2) the ability of the method to track moving targets prior to the accumulation of sufficient probability to declare a detection ("track before detect"); and (3) its ability to track multiple targets even through close approach with practically the same computational effort as for a single target.

1.0 Introduction

The essence of stochastic filtering is the computation of the time dependent probability density function for the state of an observed system by performing (linear or nonlinear) operations on an historical ensemble of observed data ([ref Bucy]). The Kalman-Bucy filter constructs the optimum estimate of the probability density in the case where the observed system is linear and the probabilities involved are Gaussian.

The optimum mathematical methods for spatiotemporal filtering have been known for some time. When the observed system is governed by linear differential equations and the noises are Gaussian, the solution is the iterative Kalman filter for discrete time samples or the Kalman-Bucy filter for continuous signals. For nonlinear systems in Gaussian noise, there are extensions of the Kalman filter. And for nonlinear systems in nonGaussian noise there is the multistage Bayesian estimator for discrete time signals or the continuous time Bayesian estimator for continuous signals.

Unfortunately, these optimum methods are well-known to be computationally intractable for all but the smallest of signals. The complexity of the iterative Kalman filter is $O(n^3)$ for general systems of order n. For practical purposes, this limits their application to signals with less than a couple dozen vector components more or less, depending on the rate at which updated estimates are required and on the available computational resources.

In particular, it is impractical to try to track a moving target in an unpreprocessed image, because the order of the system is the number of pixels in the image.

Consequently, when target recognition and tracking is the objective, the strategy typically is to reduce the order of the observed process through techniques such as model based vision, in which the time varying numerical parameters associated with certain features of the target are extracted on a frame by frame basis. In the simplest case, for example, one might use a spatial matched filter to find the range and azimuth of the centroid of some distributed target signature. These could then be tracked with a two-state Kalman filter, but most of the theoretically available gain in the spatiotemporal coherency is lost in this procedure.

2.0 Description of Method

We developed the principle of Quantum Neurodynamics (QND) to provide an integrated solution of the identification, estimation and control problems. The first objective is to provide an associative "carrier" for a large class of differential models of observed systems in order to solve the system identification problem. The second is to generate minimum squared error estimates of the observation from these differential models. And the third is to improve the model to minimize the mean squared estimation error over the accumulated ensemble of historical observations on a particular system.

2.1 The Nonlinear Schroedinger Equation

Particle physicists had an objective similar to the first of these when they sought a convenient differential model of an abstract elementary particle having wavelike properties. The result was the Schroedinger equation, which became an integral part (please excuse the pun) of the theory of Quantum Mechanics.

The Schroedinger equation has a time-dependent form, which is a linear first order partial differential equation over a complex vector space.

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = -\left(\frac{\hbar^2}{2m}\right)\nabla^2\Psi(x,t) + U(x,t)\Psi(x,t) \qquad (1)$$

Here, $\hbar$ is Planck's constant divided by 2π, i is the imaginary unit, m is the mass of the particle and ∇^2 is the Laplacian differential operator. The real-valued function U(x,t) will be described in the next paragraph. This equation has solutions $\Psi(x,t)$ in the form of complex-valued wavefunctions whose envelope (modulus squared) localizes the position of a particle in that this envelope represents a probability density function for the location of the particle in the vector space. The equation can be integrated on a discrete lattice, such as a neural network. Unfortunately, as its wavelike solutions evolve over time their particle-like envelopes disperse, i.e., the particles lose their identity. Thus the

utility of the linear time-dependent Schroedinger equation as a model of coherent particle motion only holds up over short time intervals. But this property makes it very useful for modeling increasing uncertainty about the state of a system in the absence of observations.

The scalar potential $U(x,t)$ in (1) models the force fields in which the particle defined by $\Psi(x,t)$ is constrained to move. That is, the negative gradient of the potential field at each point in the space is a force vector that deflects any nearby wave particle into regions of lower potential. Of course, since the particles defined by $\Psi(x,t)$ are actually distributed over the space as a probability density, a nonuniform gradient may tend to distort the shape of the envelope by pulling part of it in one direction and part in another. But this can be put to work to offset the dispersion of the wave-particles under conditions of observational reinforcement.

To fix the dispersion, the potential field $U(x,t)$ is supplemented with a nonlinear component to obtain the Nonlinear Schroedinger (NLS) equation:

$$i\hbar\frac{\partial\Psi(x,t)}{\partial t} = -\left(\frac{\hbar^2}{2m}\right)\nabla^2\Psi(x,t) + (U(x,t) + G(|\Psi|^2))\Psi(x,t) \quad (2)$$

Usually, $G(a)\equiv a$ so that the effect of this nonlinear component is to position a depression in the potential field that is a "shadow" of the envelope of Ψ. Therefore, whereas the dispersion tends to make the particle spread outward, this nonlinear potential shadow tends to make the particle collapse inward. When the space is one dimensional, i.e., when x is real, it is known that the resulting nonlinear Schroedinger (NLS) equation has true soliton solutions, which means, among other things, that they propagate without changing their shape (assuming that $U(x,t)=\text{const}$) and they interact with each other according to the Hamiltonian laws of particle physics. In dimension 2 and greater, the form of the nonlinearity that will produce true soliton solutions is not known, but many of the important properties of wave mechanics are known to hold for at least the nonlinearity resulting from $G(\Psi^*\Psi) = |\Psi|^2$. These include conservation of mass, conservation of momentum, and conservation of the number of particles (boson number).

2.2 Application to Neural Dynamics

Now suppose that we define a neural lattice in, say, two x dimensions, and suppose that we supply the processing elements at each node of the lattice with the nearest-neighbor communications and the instructions for integrating the NLS equation over the lattice. Then, suppose we initialize the lattice with a wave function $\Psi(x,0)$ whose envelope localizes a particle in the vicinity of the neuron at $x=x_0$, and whose wavevector corresponds to a group velocity v. If we then turn on the integrator, the wave particle will move across the neural lattice with velocity v. Moreover, if we treat the linear part

of the potential field, U(x,t), as our control input to the system, we can arrange to drive the wave particle around on just about any trajectory we like, within certain limits. (Excessive gradients will destroy the wave particles.)

Now, suppose that we code each neuron in the lattice for some "state" of the observed system. The simplest example would be to code the neuron at the lattice point x with the vector x itself, but in a more general case we might code the neurons of the lattice with synaptic patterns representing observations that result from the system being in the state x. Having done this, $|\Psi(x,t)|^2$ now represents an estimate of the probability of occurrence of the state that is *associated with the observation* coded into the neuron located at x at time t. So far this estimate is based solely on the initial condition E(x,0) and the encoding mapping. Can we somehow arrange for a continuing series of observations to drive the NLS integration so as to satisfy the requirements of stochastic filtering?

The answer, of course, is "yes". The fundamental trick is to transform the estimation error, which is in the form of a suboptimal "innovations process", into a control input in the form of the potential field U(x,t) in such a way that the resulting force field drives the wave function estimator toward the neurons whose codes are in the best agreement (in the sense of minimum squared error) with the observation. That transformation is the equivalent of the Kalman gain transformation. To implement this transformation and to accomplish other computational requirements, such as using each experiment to automatically improve the internal system model, we have designed the Parametric Avalanche neural network architecture.

2.3 The Parametric Avalanche

The Parametric Avalanche, or "PA", is a two-layer recurrent neural network architecture, as illustrated in Figure 1. The input layer is called the Novum, because it extracts the novelty from the input signal. The second layer is called the IG. The letters stand for Infinitesimal Generator, which is a technical term for the differential operator which generates the trajectory increments of a differential equation, given an initial condition. The IG subnetwork is a classifier layer that implements the Quantum Neurodynamics.

The Novum receives a spatiotemporal signal vector y(t) from the "environment" and it receives a spatiotemporal probability signal $\rho(x,t)$ from the IG. Its output is a vector of the thresholded components of the estimation error vector,

$$v(t) = y(t) - \hat{y}(t), \tag{3}$$

where

$$\hat{y}(t) = \int_{\Omega} D(x)\,\rho(x,t)\,dt \tag{4}$$

is the expectation of the observation, given the current estimate of the probability density, $\rho(x,t)$. The synapses of the Novum are represented in "transpose" form by the vector $D(x)$; that is, the i-th component of $D(x)$ is the synapse on the i-th Novum neuron that receives the output signal $\rho(x,t)$ from the IG neuron whose label is the state "x".

The IG layer receives the thresholded $v(t)$ (for simplicity, we will ignore the threshold function, as if it were linear: $\sigma(v) = v$) and fans it out to every neuron x, whose synaptic weight vector is $K(x,t)$, where it then produces the postsynaptic potential,

$$U'(x,t) = \langle K(x,t)|v(t)\rangle. \tag{5}$$

Here, $\langle a|b\rangle$ denotes the inner product of vectors a and b. Note that $U'(x,t)$ is a real scalar field over the state space $x \in \Omega$, and that if we substitute (4) into (3) and (3) into (5), then $U'(x,t)$ takes the form of the nonlinear potential field in (2), where $U(x,t) = -\langle \upsilon(x)|y(t)\rangle$ and $G(|\Psi(x,t)|^2) = \langle \upsilon(x)|\psi(t)\rangle$.

The output of the "x"-th neuron in the IG is the a-posteriori probability that the observed system is in the state $x \in \Omega$. That is, its output is $\rho(x,t) = |\Psi(x,t)|^2$, where $\Psi(x,t)$ is the wavefunction as driven by $U'(x,t)$ in response to the current observation. Note that this output is governed both by observation and by expectation, since in the latter case, the wavefunction's condition (which includes its *momentum*) prior to the current observation is the product of all the previous observations. In particular, if the observation should become occluded during some time interval, the wavefunction will continue to predict the evolution of the observed system under its own momentum, although the lack of a confirming input signal will permit any concentrations of probability (the modes of ρ) to disperse. This dispersion models the growth of

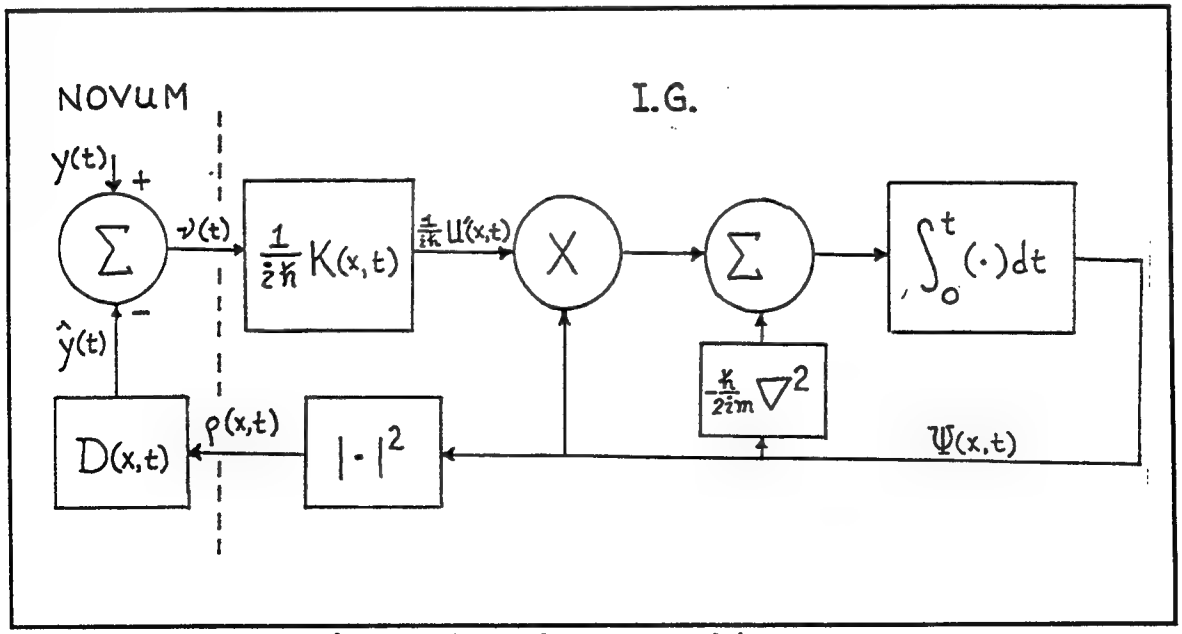


Figure 1: The PA Architecture

uncertainty, and provides an automatic mechanism for what is known in Kalman filtering as "opening the gain window" to facilitate re-acquisition of the target. These properties have all been verified by our numerical experiments.

2.4 Learning in the PA

Two distinct kinds of learning laws are used in the PA. The synapses of the Novum which receive signals being fed back from the IG employ the antiHebbian learning law,

$$\frac{\partial D(x,t)}{\partial t} = -\alpha\rho(x,t)v(t) \qquad (6)$$

The antiHebbian learning law is well known for its role in the construction of neural network novelty filters. Its is easily seen that the antiHebbian law is just a special case of the delta rule learning law, in which the desired output of the neuron is zero. This has important consequences related to error minimization, entropy maximization, and network homeostasis which are beyond the scope of this article.
The synapses K(x,t) of the IG learn by the ordinary signal Hebbian law, which is (6) without the minus sign. It turns out that whenever K(x,0) = -D(x,0) for all x and the learning rate coefficients are also the same, then K(x,t) = -D(x,t) for all $t > 0$. This reduces the memory requirement for synapses by half. It also happens to be a workable condition for the network.

When a completely novel input is presented to the Novum, i.e., an input which is orthogonal to the pattern space that is predictable by the PA, then the Novum presents a mostly intact image of that input to the synapses of every neuron of the IG. But only those neurons currently producing output are enabled to learn that pattern. Later, in a subsequent observation of that once-novel system, only the prediction error will filter through the Novum, and when it falls on the synapses of the IG neurons whose output resulted in the prediction, the error signal will add vectorially to the previously novel representation (code) to produce a corrected representation. Numerous repetitions of this process will produce an average representation of the observed pattern, weighted by the probability estimate being produced at the output of the neuron.

Although the synaptic learning laws of the PA are ultimately simple, the network nonetheless exhibits a very profound and useful "stimulus trace" mechanism due to the Quantum Neurodynamics. This is important for the use of the PA as a model of classical and extended conditioning phenomena in animals, and our research has shown that it goes far beyond simple short term conditioning to provide an explicit model for the acquisition of purely synthetic inferential knowledge by the network.

3.0 Application to Multi-Target Tracking

3.1 Detection Performance

Detection of targets is achieved by programming the synaptic templates $K(x,0)$ with the pattern that will be observed when a single target is located at the state x. In the simplest case, all templates will simply be translates of a single target signature centered on the origin, in which case the output of the Novum will in effect be convolved with that signature to produce the scalar field $U(x,t)$. More generally, however, the set of templates can be an arbitrary continuous mapping of the state space into some pattern space. We had thought at one time that these templates should be matched to the specific target signature; however, we now program the synaptic templates in the form of translates of a gaussian density representing a blurred point target. This permits tracking of arbitrary compact targets, so that target identification can be performed after the image is stabilized on the target of interest.

At the beginning of an experiment, the complex-valued wavefunction $\psi(x,0)$ is uniform over the state space. For example, if the state space consists of an 8x8 array of pixels, then $\psi(x_{i,j},0) = (0.125 , 0.0)$ for each (i,j), and the total probability, therefore, is unity. This uniformity is important, because the predetection wavefunction is like the front end of a receiver and its sensitivity is degraded by its internal noise -- especially by phase noise. For this reason, although the equations in the previous section do not show it, we have experimented with a "relaxation" term on the Schroedinger equation so that we can return the wavefunction to a quiet state after the passage of a target. Interestingly, even though soliton equations are known (cf., Newell) to be intolerant of most attempts to mess with their form, the addition of this relaxation term does not noticeably degrade the solitary wave properties of the wavepackets in our simulations.

Given the conditions above, the algorithm detects simulated targets in the input image at SNR's down to 0 dB. That is, within ten iteration cycles of the algorithm, the wavefunction's probability will have flowed toward the target's state so that the likelihood at that state is typically 0.1, versus a probability floor elsewhere of about 0.01. This performance is obtained even if the target is moving, and even if there are multiple targets in the image frame. It should be noted that the detection performance falls off if the target's velocity exceeds a significant fraction of the convolutional point spread per iteration cycle, since the "track-before-detect" capability depends on the ability of the accumulated probability at each cycle to flow into the target's potential well as it moves from cycle to cycle. Detection performance is also affected by the parameters of the Schroedinger equation, such as its mass.

3.2 Tracking Performance

Single-target tracking performance for targets at high SNR (exceeding 30 dB) whose velocity does not exceed the criteria described above yields average position estimation

errors of less than one half of the pixel spacing for simulated point targets and less than 1.5 pixels for boomerang shaped distributed target signatures. In the case of the distributed targets, the larger error is attributed not to a problem with the tracking algorithm, but to the fact that our simulator reports the "ground truth" target position as the target's center of mass, whereas its brightest point (where the probability tends to accumulate) is at the apex of the boomerang, about one pixel away. Even at 0 dB SNR, the average tracking error for point targets is less than 1.2 pixels, although at the lower SNR tracking will often be lost after an accelerated maneuver of the target (e.g., a "U" turn).

It is of interest to note that the behavior of a wave packet in our simulations as it tracks a target through a turn (at higher SNRs) reflects the known properties of quantum mechanics. That is, the acceleration through the turn results in the ejection of small pieces of the wave packet along tangents to the trajectory. These pieces are known in quantum mechanics as bremmstrahlung radiation.

Tracking of two targets is accomplished with no more computational effort than is required for tracking of a single target (not including any post processing). What is especially important about multitarget tracking is the performance of the algorithm during times when two targets are in close proximity to each other. Kalman filtering methods fail and the multi-Gaussian methods require special intercession during such times. However, our QND wave packets exhibit soliton-like characteristics in their pairwise interactions. (We have not tested interactions of more than two particles.) That is, the wave packets will merge, exchange some momentum and some probability, and emerge with their particle identity intact, continuing to track the two targets. Of course, there is no way to preserve the identity of the individual targets after the close approach, unless one is exploiting other characteristics of the signatures for identification in some post processing stage. (We are exploring the latter possibilities in an Army-sponsored research project.)

4.0 Application to Control of Nonlinear Systems

In a NASA-sponsored project, we have employed a simplified version of the Parametric Avalanche as an observer in a state feedback controller for nonlinear systems. We will only describe those results briefly in this paper.

Our controller consists of a network of one-dimensional PA control modules, called PACM's. Each PACM has only a single pixel in the Novum, and the IG is a one dimensional array of 100 neurons arranges in a circle. The propagation of wavepackets is only simulated in the PACM, i.e., by translating an envelope around the circle, rather than by integrating the Schroedinger equation. We are applying these controllers to the nonlinearly controllable stabilization problem for tethered satellites. Here, however, we shall show only the performance that has been obtained using two PACM's to solve the constrained "broom balancing" problem.

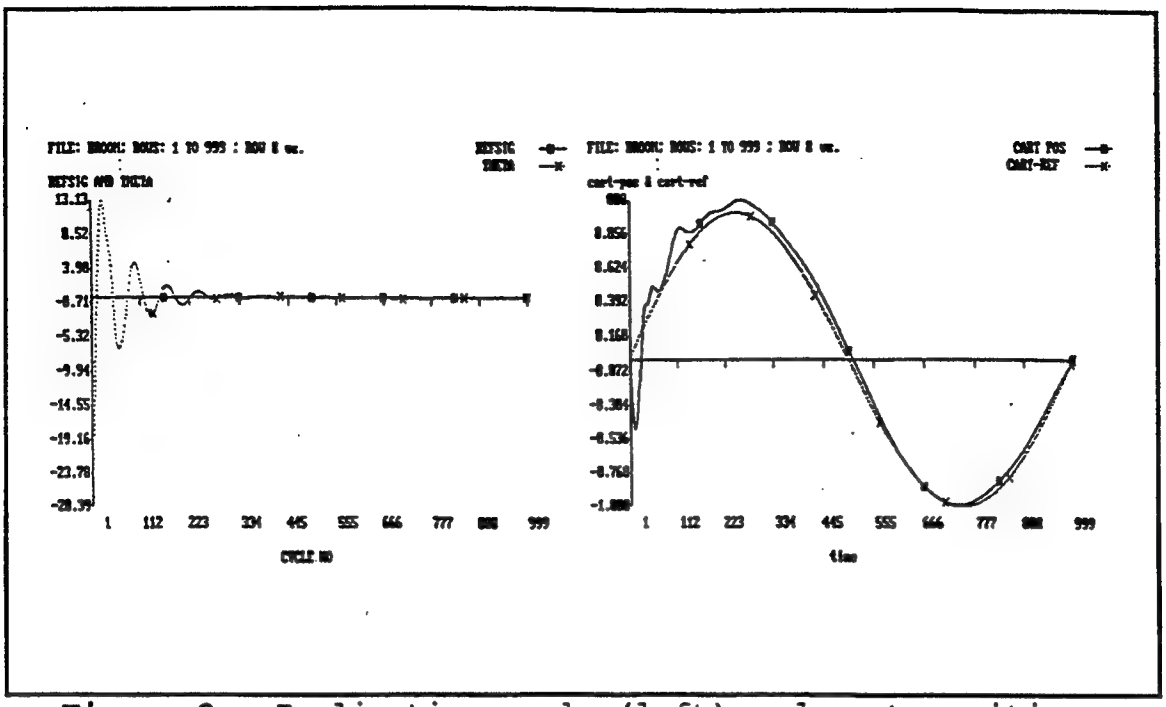


Figure 2. Inclination angle (left) and cart position.

Figure 2 shows the result of a numerical experiment in which the initial angle of the pendulum is -30 degrees, and the cart is commanded to track a moving reference point on the driveway. The reference point moves first north from the origin, then south through the origin and back again in a sinusoidal displacement profile. The inputs to the two PACM's consist of the inclination angle of the pendulum, the x position of the cart, and the commanded reference position for the cart.

The inclination angle θ is supplied to the input of the first of two PACMs, and the x position is supplied to the input of the second PACM. The commanded position is supplied as the reference input to the second PACM, while the commanded angle for the first PACM is supplied by the Novum output of the second PACM. In this way, the position is controlled by using the inverted pendulum as a "joystick" command for the first PACM to try to maintain by accelerating the cart in the appropriate direction. The control output for driving the cart is the Novum output (appropriately scaled) of the first PACM.

Similar results are obtained when random impulse noise is added to the angular acceleration of the pendulum.

5.0 Conclusion

This computing architecture, developed with the support of the Naval Surface Warfare Center and other government agencies, represents a significant breakthrough in target detection and tracking technology. It also represents an important new model of cognitive

systems that will lead to highly competitive intelligent computing architectures. We expect that new computing machines based on these methods will be vastly different from existing computers that are based on symbolic and heuristic manipulations. That is, these new machines will employ a combination of "cloned instinct" from previously trained units, and environmental adaptation to perform complex exploration and control tasks in the service of generally stated mission constraints.

Exploitation of our competitive technological advantage will require a focused application of resources to resolve a number of issues ranging from theoretical to hardware engineering. We are in a position much like in World War II when the nation was presented with the choice to be first in the development of atomic weapons or to suffer the consequences of being late. Now, as then, the technology is revolutionary, it is volatile and our advantage is only as great as our vision and our commitment to winning. Now, as then, the consequences of being second are unthinkable.

6.0 References

1. Ablowitz, M., and Segur, H., Solitons and the Inverse Scattering Transform, SIAM, Philadelphia, 1981.

2. Ames, W.F., Numerical Methods for Partial Differential Equations, Academic Press, New York, 1977.

3. Anderson, B.D.O., and Moore, John B., Optimal Filtering, Prentice-Hall, Englewood Cliffs, NJ, 1979.

4. Bialynicki-Birula, I., and Mycielski, J., "Nonlinear Wave Mechanics", Annals of Physics 100, 62-93 (1976).

5. R.S. Bucy, "Linear and Nonlinear Filtering", Proc. IEEE, 58, No. 6, June 1970.

6. Dawes, R.L., "Quantum Neurodynamics and the Parametric Avalanche", Technical Report MRC-NASA-89004, Martingale Research Corporation, Allen, TX, August, 1989.

7. Dawes, R.L., "Quantum Neurodynamics: A New Neural Network Paradigm for Knowledge Representation, Cognition, and Control", Technical Report MRC-NSF-89003, Martingale Research Corporation, Allen, TX, July, 1989.

8. Dawes, R.L., "Adaptive Control and Stabilization with the Parametric Avalanche", Technical Report MRC-ARDEC-89002, Martingale Research Corporation, Allen, TX, April, 1989.

9. Dawes, R.L., "Automatic Target Recognition and Tracking with the Parametric Avalanche", Technical Report MRC-AFWAL-89001, Martingale Research

Corporation, Allen, TX, March, 1989.

10. Dawes, R., "Theory and Feasibility of the Parametric Avalanche for Image Understanding and Control", Technical Report MRC-DARPA-88-003, Martingale Research Corp., Allen TX, March 1988.

11. M. Goldman, "Strong turbulence of plasma waves", Reviews of Modern Physics, Vol. 56, No. 4, October 1984.

12. M. Goldman, "Langmuir wave solitons and spatial collapse in plasma physics", Physica 18D, 1986, 67-76.

13. Grossberg, S., "Some Networks That Can Learn, Remember, and Reproduce Any Number of Complicated Space-Time Patterns, II", Studies in Appl. Math., Vol XLIX, No. 2, June 1970.

14. Ho, Y.C., and Lee, R.C.K., "A Bayesian Approach to Problems in Stochastic Estimation and Control", IEEE Trans. Automat. Contr., vol. AC-9, pp. 333-339, Oct. 1964.

15. Kailath, T., "An Innovations Approach to Least-Squares Estimation, Part I: Linear Filtering in Additive White Noise", IEEE Trans. Automat. Contr., vol. AC-13, pp. 646-655, Dec. 1968.

16. Kailath, T., "The Innovations Approach to Detection and Estimation Theory", Proc. IEEE, vol 58, No. 5, May 1970.

17. Kallianpur, G., Stochastic Filtering Theory, Springer-Verlag, 1980.

18. Kalman, R.E., Falb, P., and Arbib, M., Topics in Mathematical System Theory, McGraw-Hill, New York, 1969.

19. Kohonen, T., Self-Organization and Associative Memory, Springer Verlag, New York, 1984.

20. Krishnan, V., Nonlinear Filtering and Smoothing: An Introduction to Martingales, Stochastic Integrals and Estimation, John Wiley & Sons, New York, 1984.

21. Newell, Alan C., Solitons in Mathematics and Physics, SIAM, CBMS-NSF No. 48, Philadelphia, 1985.

22. Sorenson, H.W. (Ed.), Kalman Filtering: Theory and Application, IEEE Press, New York, 1985.

23. Taniuti, T., & Nishihara, K., Nonlinear Waves, Pitman Publishing Inc., Marshfield, MA, 1983.

24. Yashin, A., "Continuous-Time Adaptive Filtering", IEEE Trans. on Automatic Control, Vol AC-31, No. 8, August 1986.

A Neural Network and Expert System Approach to Multiple Target Recognition

Alper K. Caglayan and Paul G. Gonsalves
Charles River Analytics Inc.
55 Wheeler Street
Cambridge, MA 02138
(617)491-3474

Abstract: In this paper, we present a hybrid artificial neural network (ANN)/knowledge base (KB) approach to multiple target recognition (MTR). Specifically, we develop a hybrid MTR architecture composed of ANN and KB classifiers and decision makers, and conventional signal processing and probabilistic target tracking algorithms. Our approach is based on the use of invariance based ANN input features, the selection of training data using distance metrics, and the interpretation of ANN outputs using knowledge base inference.

1.0 Introduction

Naval air defense refers to the protection of sea-based assets from airborne threats. These systems face an ever increasing challenge due to newer more sophisticated and capable aircraft and missile systems. In addition, the complexity and time constraint aspects during tactical situations present even more requirements on current shipboard systems. The need for a concise and rapid assessment of threat status and determining the associated response is a major task for a ship commander. In response to these requirements, significant effort has been made in countering this threat with advances in sensor capabilities, sensor fusion, and multiple target tracking. This rapid evolution must be met with concomitant advances in the recognition and classification of tracked targets or more commonly referred to as multiple target recognition (MTR). Multiple target recognition systems are real-time information management systems which process and assess multi-sensor data, and present the best options to a decision maker.

The major difficulties in target recognition problems are the uncertainty of signal origin, the correlation of the multi-sensor data, the fluctuations in signal-noise-ratio, interference from sources other than the targets intended, loss of target contact with the associated fades and gaps in the measurement data, execution of various maneuvers by the tracked targets, and the complexity of the signal processing algorithms relating the raw sensor data to target state information. The most important characteristic of sensor data in multiple target tracking is the uncertainty of its origin. This uncertainty can arise not only from clutter, interference and multipath effects, but also from multiple targets in the same neighborhood. Research over the last two decades has been driven to find a computationally tractable way of incorporating measurements of uncertain origin into existing tracks. These algorithms can be roughly divided into two groups: non-Bayesian and Bayesian. Non-Bayesian algorithms estimate target states conditioned on the correctness of their decisions on target tracks. Hence, the resulting target state estimates and covariances do not take into account for the possibility of incorrect decisions. On the other hand, Bayesian algorithms for tracking problems account for the probability that the

decisions may be incorrect. As discussed by Bar-Shalom (1978) and by Chang and Tabaczynski (1984), while there is no tracking algorithm capable of working reliably in an arbitrarily dense target environment, there are algorithms which can provide acceptable tracking performance for a limited target density.

A hybrid artificial neural network (ANN) and expert system approach can significantly enhance the performance of current multiple target tracking systems. Artificial neural networks produce a nearest neighbor classifier. Conceptually, artificial neural networks are applicable to multiple target tracking problems by learning the spatiotemporal attributes of target trajectories and classifying multi-sensor data. For instance, a multilayered perception has been trained to model optimum guidance trajectories for the aeroassisted orbital plane change scenario in Caglayan and Allen (1990). The use of neural networks in scan-scan correlation in multi-target tracking has been investigated by Kuczewski (1989). In Anderson et al. (1990), a neural network-based system is presented that is capable of clustering and identifying radar emitters. A discussion of the potential impact that neural network technology may have on target recognition systems is found in Roth (1990). An artificial neural network approach to multiple target tracking offers several advantages including a rapidly adaptable on-line solution via new models and learning algorithms (e.g., addressing the need for adaptation to target and environment changes), implementation efficiency on the emerging neural computers, and off-line productivity improvement by reducing trial and error.

While neural networks can play a significant role in multiple target tracking, a pure neural net approach would mimic only the right hemisphere functions of the human brain. In order to incorporate formal knowledge (i.e., probabilistic algorithms) and qualitative reasoning of human experts requires the addition of an expert system approach emulating the left hemisphere function of the human brain. As an example of encoding domain specific knowledge, Lin and Chen (1991) demonstrate the use of a knowledge based system for a land-based automated air defense system.

In this paper, we discuss relevant issues pertaining to hybrid system development. We also present two competing hybrid system architectures for the MTR problem. The two architectures are implemented using our commercially-available hybrid system development tool NueX. The system is then demonstrated under several specific scenarios involving variation in sensor noise levels and object density.

2.0 Artificial Neural Networks

The on-line classification performance of an artificial neural network (ANN) is duly influenced by the selection of input features and training data. In most problems, the determination of ANN outputs is fairly straight forward since the problem requirements usually dictate the form of decision to be computed by a neural network. In contrast, the determination of the ANN inputs in MTR is not that simple since there are usually an abundance of input data which could be used including different types of sensors, spatially varying characteristics of sensor data, temporal variation of sensor data, and others. Hence, MTR neural network design problem requires the identification of pertinent inputs and the reduction of input data to reduce the dimensionality of

the feature space. Also, the abundance of data present in an MTR application for use in training necessitates reduction in data not only for reducing the training time, but also for finding conditions such as conflicting data which inhibit learning. Here, we present a unified approach to determine relevant input features and to training data selection geared at finding conflicting data, redundant data, data clusters, and orthogonality of data.

2.1 Invariant Feature Extraction

The identification of the ANN input types can best be determined from an analysis of the physics of the problem. The data reduction involves the identification of relevant features affecting the decision. The feature selection objective is to maximize the similarity of objects in the same class while maximizing the dissimilarity of objects in different classes. We believe that the data reduction process can be encapsulated into rules for certain types of ANN problems. These rules can be obtained, for instance, by invariance considerations. Invariance attributes for a neural net have been traditionally imposed by structural constraints (e.g., symmetric network weights). The major problem of this approach is the prohibitively high number of nodes required for structural invariance. A more practical approach to invariance is to extract features that are invariant under transformations of an input.

Other approaches to invariance include training based and constraint based techniques. In training based invariance, the neural network is presented with the different transformations of the same input and asked to derive the transformation invariance. For instance, in backpropagation algorithms, if the number of nodes in a hidden layer and biases are variables of optimization, then the ANN essentially determines the hyperplanes necessary for the classification decision. The main drawback to this approach is the increased input data size and training time.

The constraint based approach is a general method for obtaining invariant feature spaces by using constraints (Barnard and Casasent (1991)). To outline this approach, consider the following. Let S denote a set of measurements to be classified invariantly. Let the transformations affecting the required invariance be denoted by $T\alpha$ where (α is the set of parameters specifying the transformation:

$$T\alpha: S \rightarrow F \text{ with } f=T\alpha(s) \text{ for min F and min S} \qquad (1)$$

where F is the feature space, f and s are elements of F and S. Defining the trajectory, $\Omega(s)$, in the feature space

$$\Omega(s) = \{f:f=T\alpha(s) \text{ for some } \alpha\} \qquad (2)$$

$\Omega(s)$ denotes equivalence classes in the feature space. Choose a single point on this trajectory to compactly describe the trajectories. This is called the characteristic point of the feature trajectory:

$$C_i(f_o) = 0 \text{ for all i} \qquad (3)$$

the invariant feature space is then characterized by finding all satisfying

$$C_i(T\alpha(s)) = 0 \text{ for all } i \tag{4}$$

Bamard and Casasent demonstrates the use of constraint based invariance in determining neural net features in Bamard and Casasent (1991). They show that features based on translations, scaling and rotation constraints of tank target images generated by laser range sensor produces classification performance with 95% level performance. As indicated in their research, the neural classifiers do not offer significant performance improvement over other classifiers for the simple problem considered, while indicating greater improvement in performance with more target classes and types. We believe that invariance based features can be applied successfully to MTR to resolve the problems associated with features dependent on aspect angle, and target states.

2.2 MTR Training Data Preparation

The feedforward multi-layered network is a function mapping the n-dimensional vector space $[0,1]^n$ into the m-dimensional vector space $[0,1]^m$ where [0,1] is the closed interval between 0 and 1 on the real line, n is the number of network inputs and m is the number of network outputs. In order to analyze the training data for conflicting and redundant input/output pairs, we need a distance measure between two input vectors, between two output vectors, and between two input/output pair vectors. There are a number of distance metrics derived from the following norms:

$$d_s(x(k), x(j)) = |x_1(k) - x_1(j)| + |x_2(k) - x_2(j)| + \ldots + |x_n(k) - x_n(j)| \tag{5}$$

$$d_m(x(k), x(j)) = \max |x_i(k) - x_i(j)| \tag{6}$$

$$d_2(x(k), x(j)) = ((x_1(k) - x_1(j))^2 + \ldots + (x_n(k) - x_n(j)^2)^{1/2} \tag{7}$$

where $x_i(k)$ is the i'th input in the k'th input/output pair. In the following, we use a generic distance d(x,y), although we have found the max-norm induced distance metric dm(x,y) to be the most convenient in our studies.

2.2.1 Conflicting Data

Conflicting data are two or more input/output pairs where two similar inputs are mapped into dissimilar outputs. These input/output pairs violate the definition of a function where one input is mapped to one and only one output, thus making the training of a network impossible. In analyzing the training data to find conflicting input/output pairs, the following function s:

$$s(k,j) = \frac{d(y(k), y(j))}{d(x(k), x(j))} \tag{8}$$

gives a measure for the degree of conflict between the input/output pair {x(j),y(j)} and {x(k),y(k)}. The higher the value of s, the higher the degree of conflict between the two input

output pair vectors given that the input vectors are nearly identical.

The procedure for determining conflicting data is illustrated in Figure 1 for training data taken from one of the ANN classifiers used in this study. The network has two dimensional input nodes and a scalar output which is either inhibited or activated. Shown in Figure 1 is the measure of confliction between input/output pairs (IOPS) as given by the function s sorted in ascending order. Two regions of s occur: one for which s is zero and a non-zero region. The former corresponds to IOPs having the same output while the latter is for differing outputs. Conflicting data can only occur with differing outputs, so it suffices to investigate the IOPs associated with high values of s, say for this situation a s value greater than six. These IOPs would be then candidates for elimination from the training database.

2.2.2 Redundant Data

Redundant data are two or more input/output pairs where two nearly identical inputs are mapped into similar outputs. While this condition will not prohibit learning, it would adversely affect training time. In order to analyze the training data to find redundant input/output pairs, we need a distance measure on the cartesian product X x Y, which is the n + m dimensional vector space. We can define distance measures on X x Y in terms of the distance measures on X and Y by:

$$d_s(z(k), z(j)) = d_x(x(k), x(j)) + d_y(y(k), y(j)) \quad (9)$$

$$d_m(z(k), z(j)) = \max_{x,y} \{d_x(x(k), x(j)), d_y(y(k), y(j)\} \quad (10)$$

$$d_2(z(k), z(j)) = (d^2_x(x(k), x(j) + d^2_y(y(k), y(j))^{1/2} \quad (11)$$

where z(k) and z(j) are the k'th and j'th input output pair vectors defined by:

$$z(k) = \{x(k), y(k)\} \quad (12)$$
$$z(j) = \{x(j), y(j)\} \quad (13)$$

We have found the max-norm induced metric to be the most convenient. Any of these metrics can be used to find almost identical data (where the distance between the input/output pair vectors would be nearly zero).

An example of redundant data is illustrated in Figure 2, again taken from the same ANN classifier as for the conflicting data. Shown in Figure 2 is the measure of redundancy between input output pairs (IOPs) as given by max norm metric of (10). The smaller the metric value, the greater the amount of redundancy between the associated IOPS. Again, two regions occur: one for which the metric is nonconstant and a constant region with a metric value of 0.8. The former corresponds to IOPs having the same output while the latter is for differing outputs and thus lead to a higher metric value and thus are not redundant. IOPs with a metric value below 0. 1 would be candidates for elimination due to redundancy in this case.

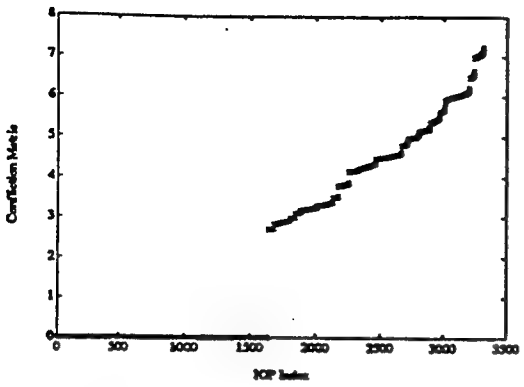


Figure 1: Conflicting Training Data

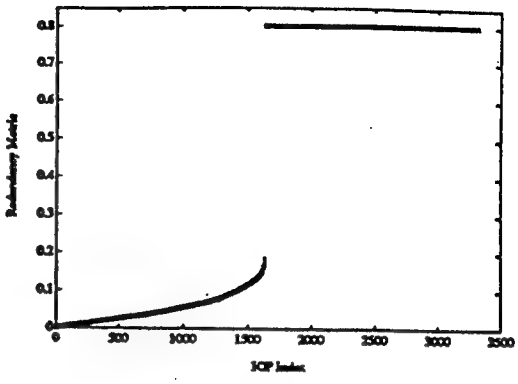


Figure 2: Redundant Training Data

2.2.3 Clustered Data

Clustered data are the subsets of the training data, where the input/output pairs in each cluster subset are redundant. Data clusters can be found using the notion of formalized equality sets in Lorczak et al. (1989) with a distance metric on X and Y. According to this approach, a subset S of the input/output pairs is a formalized equality set if

$$d(z(k), z(j)) \leq \epsilon \tag{14}$$

for any two input/output pairs z(k) and z(j) in S. Here, ϵ is a small number dictating the desired granularity. This clustering analysis can be performed to select a median input output pair from each cluster class.

A sample clustering is shown in Figure 3, again for the same ANN classifier. Two data clusters exist for the two output states. The inhibited output state cluster data points are denoted by a "o", whereas the activated output state data points are denoted by "+". The training of this network could be reduced by selecting the boundary IOPs from these two clusters and eliminating the rest of the pairs within the clusters.

2.2.4 Data Orthogonality

Another desirable attribute of the training data is to have input/output pairs as orthogonal as possible. In order to do this analysis, we need an inner product on X x Y. We can define such an inner product by:

$$\langle z(k), z(j) \rangle = 0.5\ (\langle x(k), x(j) \rangle + \langle y(k), y(j) \rangle) \tag{15}$$

where

$$\langle x(k), x(j) \rangle = (x_1,(k)x_1(j) + \ldots + x_n(k)x_n(j))/n \tag{16}$$

and

$$\langle y(k), y(j) \rangle = (y_1,(k)y_1(j) + ... + y_m(k)y_m(j)/m \quad (17)$$

Such an inner product would yield a range of [0,1], where a value of zero would indicate orthogonality and a value of 1 would indicate redundancy.

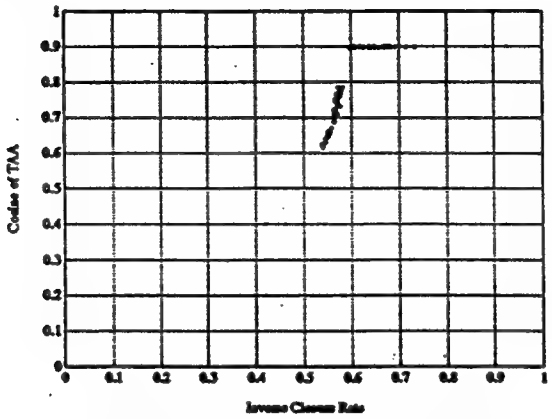


Figure 3: Training Data Clusters

3.0 System Implementation

Figure 4 presents an overview block diagram of the hybrid MTR system implementation. The implementation is broken down into two specific areas: the external modules that provide off-line generation of trajectory sensor data and the MTR hybrid system. The external modules consist of target and ownship models that are used to generate object/ownship position and velocity states and a radar sensor subsystem block that generates measurements of target relative states

Two hybrid architectures have been implemented. The first hybrid system consists of a target tracking estimator which computes radar measurement residuals assuming the target is executing a hostile guidance law, an ANN classifier driven by the measurement residuals, a knowledge base (KB) classifier processing the sensor measurements directly, and a knowledge base decision maker integrating the ANN and KB classification decisions. The second hybrid system replaces the estimator and the residual-based ANN classifier with a sensor feature extraction algorithm based on invariance and an ANN classifier driven by the sensor features, respectively. The first hybrid system's performance is primarily influenced by the tracking algorithm error under modeling errors, typical of probabilistic tracking algorithms. On the other hand, the second hybrid system uses a more direct neural network approach where the analytic algorithms are used only for feature extraction without making any modeling assumptions about the target dynamics.

3.1 Target and Ownship Modules

The target and ownship modules simulate multiple hostile and friendly targets and the tracking platform. The modules generate time histories of target and platform position, velocity, and acceleration. The modules support three modes of trajectory generation operation: 1) trajectories

employing a turn-to-heading intercept guidance; 2) trajectories employing a line-of-sight (LOS) intercept; and 3) trajectories using zero acceleration/constant velocity.

3.2 Radar Sensor Module

The MTR sensor path is driven by the target and ownship position and velocity vectors, $\mathbf{r}_t$, $\mathbf{v}_t$, and $\mathbf{r}$, $\mathbf{v}$, respectively. Their difference yields an ownship relative set of position and velocity states ($\mathbf{r}_r$ and $\mathbf{v}_r$). These relative states are then transformed from cartesian into spherical coordinates. The resulting six dimensional spherical states (range, elevation, azimuth, range rate, elevation rate, and azimuth rate are then corrupted with noise to simulate radar sensor measurements. The range and range rate signals are corrupted via multiplicative noise. The angles and angular rates are corrupted with additive noise. The six independent noise sources are uncorrelated sequences with zero-mean normal distribution. Covariances are chosen to yield +/-1 degree errors in angular measurements and +/-. 1 degree/s errors in angular rates. For the range and range rate measurements, the unit variance noise source are multiplied by the current range/range rate and by a multiplicative factor. The values chosen correspond to a signal to-noise ratio (SNR) of 25 db.

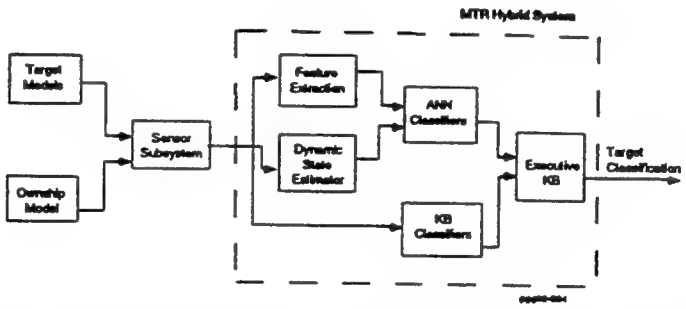


Figure 4: Overview Block Diagram of MTR System Implementation

3.3 MTR Hybrid System

The MTR hybrid system provides for the on-line classification of targets based on sensor measurements. It uses both knowledge bases and neural networks to perform the classification function. A functional block diagram of the hybrid system is shown in Figure 5. As shown, an executive knowledge base controls the overall classification process by interrogating in parallel one of two neural networks and another knowledge base. The user specifies which of the two networks to use. The two networks have the same topology but use different input features, i.e., sensor-derived invariance-based features or estimator generated velocity measurement residuals. Results of the neural network and the classification knowledge base are integrated by the executive knowledge base, which then makes the final decision on target classification. In the rest of this section we further describe the individual elements of MTR Hybrid System and the implementation of the system within our hybrid development tool NueX.

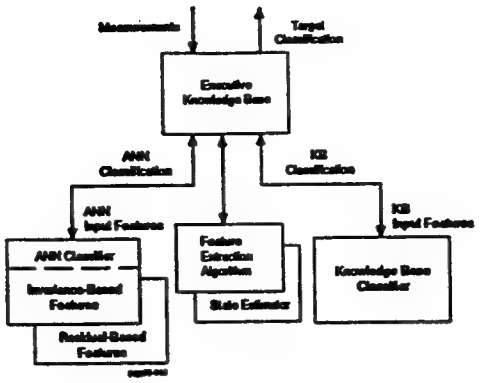


Figure 5: Functional Block Diagram of the Hybrid System

3.3.1 Dynamic State Estimator

The dynamic state estimator is based on the extended Kalman Filter algorithm. We apply the extended Kalman Filter formulation to the estimation problem at hand. We use a relative coordinate system in which only the target is accelerating. The system dynamics model is then given by

$$\dot{\mathbf{r}} = \mathbf{v}$$
$$\dot{\mathbf{i}} = \mathbf{a}(t) + \mathbf{w} \qquad (18)$$

where r and v are the relative three-dimensional cartesian position and velocity state vectors between the target and the ownship. The input term a(t) is the relative acceleration vector. If ownship is stationary or non-accelerating, a(t) becomes the target's acceleration. In order to drive the Kalman Filter, we make the *assumption* that all targets are using a turn-to-heading intercept guidance law. Therefore, at each time step, and for each target, a(t) is calculated using current state estimates and then fed to the state estimator to generate new estimates of target relative position and velocity. Hence, the measurement residuals of this tracking filter will be zero-mean when the target is hostile and executing the assumed intercept law. In contrast, the measurement residuals will be non-zero mean when the target is friendly.

3.3.2 Feature Extractor

Feature extraction allows for the reduction and transformation of the data input set into pertinent features that the ANNs can use for classification purposes. For the problem at hand the data input set consists of sensor-derived measurements of position and velocity in spherical coordinates, and filter generated state estimates of position and velocity in cartesian coordinates. From our analysis, two neural network input features were derived: estimator generated target velocity measurement residuals and sensor derived target features based on invariance

The first set of features is based on relative two-dimensional velocity measurement residuals in

the self-centered cartesian frame of reference given by:

$$\tilde{v}_x(k) = v_x(k) - \hat{v}_x(k \mid k-1)$$
$$\tilde{v}_y(k) = v_y(k) - \hat{v}_y(k \mid k-1) \qquad (19)$$

where v_x is the x-velocity pseudomeasurement, $v(k \mid k-1)$ is the single stage prediction of the x-velocity pseudomeasurement generated by the target state estimator and $v_x(k)$ is the corresponding measurement residual at frame k. These measurement residuals scaled by the standard deviation of the measurement noise, and averaged over a moving window are used as feature inputs into an ANN.

The second set of features is derived directly from the sensor measurements without filtering using invariance considerations. These are the cosine of Target Aspect Angle and the inverse closure rate as given by:

$$\cos(TAA) = \frac{\underline{v} \cdot \underline{r}}{|\underline{v}|\,|\underline{r}|}$$

$$\tau = \frac{\dot{P}}{P} \qquad (20)$$

where TAA is the Target Aspect Angle (i.e., the angle between the LOS vector and the target velocity), v is the target velocity vector, r is the target position vector, and - denotes the vector dot product. The cosine of the Target Aspect Angle is given by the dot product of the position and velocity unit vectors. τ is the inverse of the target closure rate where the closure rate is given by range divided by the range rate.

The ANN features are selected based on invariance considerations depicted in Figures 6 and 7. Figure 6 shows a typical set of target maneuvers which would yield the same feature based on the Target Aspect Angle. Figure 7 shows a typical set of target maneuvers which would yield the same ANN input feature based on the closure rate. Thus invariance property is important both in minimizing the training data and in classifying the target behavior.

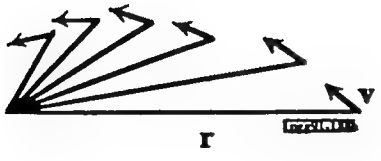


Figure 6: Target Maneuvers with Identical Aspect Angles

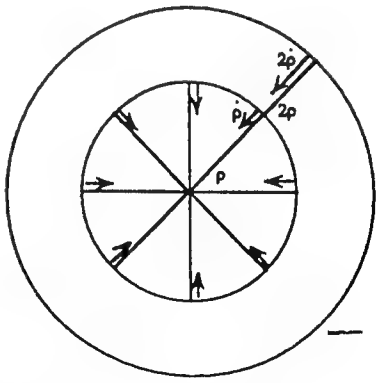


Figure 7: Target Maneuvers with Identical Closure Rates

3.4 Neural Network Classifiers

In the section we detail the specification and training of the two neural networks used in the hybrid MTR system: the network using the estimator residual-based features, and the network using the sensor derived invariance-based features. For both networks we employ a feedforward backpropagation network with an architecture as shown in Figure 8. The architecture consists of: processing the input features through delay operators (denoted by D where D(x(k)) = x(k- 1)) to provide a moving window average of the ANN input features; a hidden layer with ten nodes; and a single output signal which is also processed through delay operators to provide a moving window average for target classification. The moving window average is based on four measurements. The moving windows are used to introduce filtering to enhance the ANN target classification performance. The ANN topology is fully-connected. The number of hidden nodes is determined by training performance and by the number of input/output pairs presented.

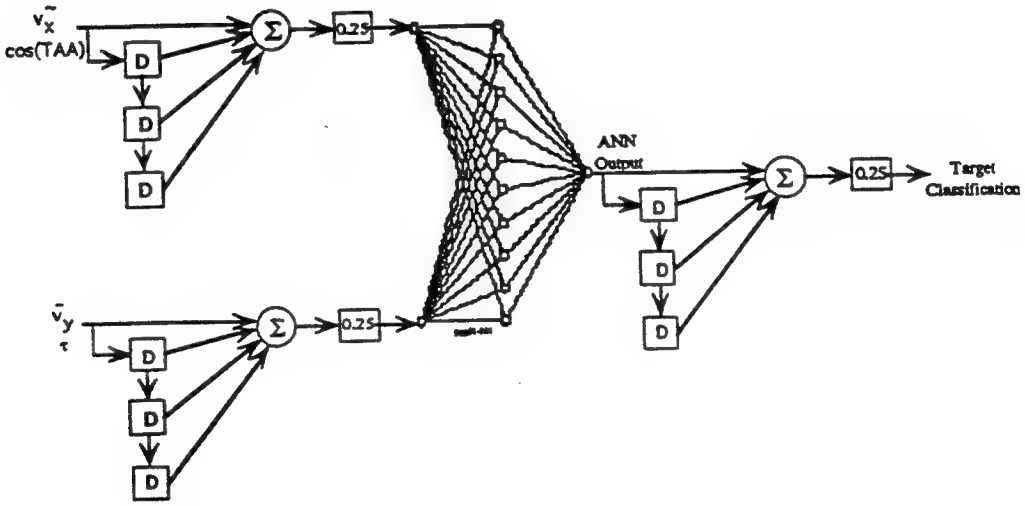


Figure 8: ANN Classifier Architecture

The next step in neural network classifier specification is the training of the networks. Training involves presenting the network input/output pairs and updating the network parameters (weights and biases) via the backpropagation learning algorithm until an acceptable level of error between desired and network outputs is achieved. At that point, the network parameters are frozen and the network can evaluated for its on-line classification performance. Training data for both networks were generated for both hostile and non-hostile targets. Both networks were trained such that the output node was activated for a hostile trajectory and inhibited for a friendly trajectory.

After training, both networks with their associated parameters were then analyzed by investigating the input/output relationships or equivalently looking at the decision space of the networks. Figure 9 shows the decision space for the trained residual-based network. The v_x, v_y region highlighted with a "+" symbol corresponds to the inputs that will activate the output node, or equivalently a hostile classification. The region denoted with the "o" symbol corresponds to the input that will inhibit the output node, unknown classification. As shown, a hostile classification results when both v_x and v_y are within the +/-3 standard deviations region with a few point points stretching to 4 standard deviations. A friendly classification occurs when one or both of the input features exceed 5 or 6 standard deviations.

The invariance-based neural network was trained using trajectory data. For training purposes, two hostile trajectories and two friendly target trajectories were used. This network was able to train to lower error levels than the residual-based network because the region in the decision space where hostiles and friendlies are determined is more easily defined due to invariance. This is verified by inspecting Figure 10 which shows the decision hyperplanes in the feature space. The points denoted by "*" correspond to an unknown classification. The region within these points results in a hostile classification. The region outside corresponds to a friendly target classification. This example clearly shows that the neural networks can replace the binary hypothesis, sequential probability ratio, and multiple hypothesis tests in MTR functions. Furthermore, the ANN based approach is basically a thresholdless approach where the decision hyperplanes (lines in this example) in the feature space are automatically generated by the network learning process.

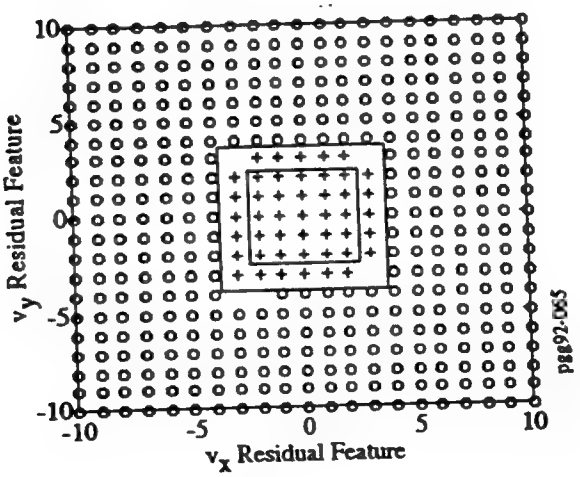


Figure 9: Residual Network Decision Space

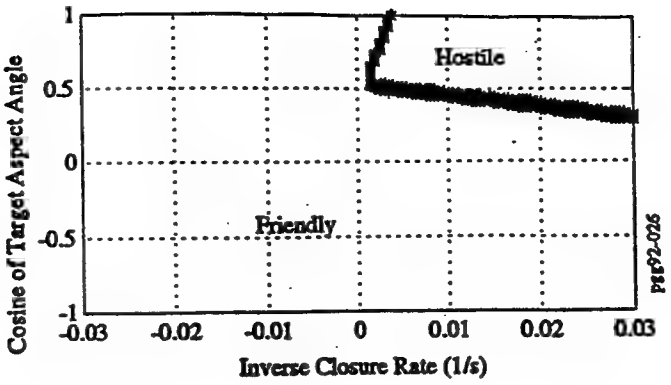


Figure 10: Decision Hyperplanes in Invariance Feature Space

3.4.1 Classification Knowledge Base

The classification knowledge base provides the functionality to encode domain specific expert reasoning to the problem of target classification. The knowledge base incorporates this domain specific expertise into rules which are composed in an IF-THEN structure. The rules are implemented to perform target classification using sensor derived target data input features. The specific rules used in the knowledge base classifier are documented in Figure 11. The figure lists the IF and THEN parts of the rules in order of firing sequence priority, i.e., in the order which rules would be fired first. The rule set consists of nine rules altogether. The first five rules declare the target friendly or hostile based on target range, range rate, and the cosine of the Target Aspect Angle (TAA). The cos(TAA) provides an indication of the target's heading, whereas range and range rate provide indicators of proximity and closure rate. The specific numbers used in the conditional part of these rules are derived from an assumption of how fast the target can reach a danger region assumed to be within a 30 nm range. For instance, in rule 1, the current knowledge base output is set to hostile if target range is between 55 and 60 nm, its heading is generally towards self, and it can reach the danger region within at least 5 minutes. For closer targets, this time to danger region is increased because proximity becomes more of a concern. The current knowledge base output is set to friendly when targets appear to be moving away from self or acting in a non-hostile manner. Rules 6 and 7 account for the possibility that a hostile target could be performing a maneuver when moving away from self. That is, a hostile target could be turning in order to acquire ownship. The manner in which these rules make this determination is by inspecting a knowledge base feature called maneuver-detect based on the time rate of change of the cos(TAA). The final two rules of the classification knowledge base assert decisions passed to the executive knowledge base. If the target has been asserted hostile three of the four last times by the knowledge base as determined in rules 1 through 5, the target is classified a hostile. If it has been classified friendly three out of the last four times and it is not maneuvering, the classification knowledge base classifies the target as a friendly. If neither of these rules apply, no classification is made and the target is still considered unknown.

Rule Number	IF	THEN	Rule Firing* Priority
1	no target within sensor range (60 nm) & range < 55 nm & range rate < -700 ft/s & cosine(TAA) > 0.7	assert current target status is hostile	High
2	range < 55 nm & range ≥ 45 nm & range rate < -500 ft/s & cosine(TAA) > 0.8	assert current target status is hostile	High
3	range < 45nm & range ≥ 30 nm & range rate < -100 ft/s & cosine(TAA) > 0.9	assert current target status is hostile	High
4	range < 55 nm & range rate > 0 ft/s & cosine(TAA) < 0	assert current target status is friendly	High
5	range < 45 nm & range rate > 0 ft/s & cosine(TAA) < 0.5	assert current target status is friendly	High
6	-4 < maneuver_detect < 4	assert target is not manuvering	Med
7	maneuver_detect > 4 or maneuver_detect < -4	assert target is manuvering	Med
8	target has been asserted hostile three of last four times	assert target is hostile	Low
9	target has been asserted friendly three of last four times & target is not maneuvering	assert target is friendly	Low

pgg92-018

*ordering precedence for firing of rules

Figure 11 Clasification Knowledge Base

3.4.2 Executive Knowledge Base

The overall target classification decision-making is performed in the executive knowledge base. This knowledge base interrogates one of two ANNs (either the residual-based features or invariant-based features network) and the classification knowledge base and uses ANN, KB, and other information to make a final decision on target status. The specific rules implemented are listed in Figure 12 in order of rule firing priority.

The three rules with the highest rule firing priority perform target classification based on information other than in the classification knowledge base or the ANNS. Specifically, these rules deal with information the ship commander may have from other sources such as Identification Friend or Foe (IFF) systems or making the determination that if a target has radar lockon on ownship, then it is hostile. The third rule accounts for targets that are initially within sensor range, but than go out of range and are never detected again. These targets are assumed friendlies due to their non-hostile trajectories (i.e., their heading directly away from ownship). Rule 4 has the next highest priority level. It basically serves as a flag to determine if the current target has been classified and if it is within sensor range. If both conditions are true, then the classification procedure for the target will be implemented. This procedure is set forth in rules 5 through 7. Here, the executive knowledge base interrogates the classification knowledge base and the selected ANN. Rules 8 and 9 take the average of the last four outputs of the selected ANN and make a classification decision according to the ANN. The overall MTR system target

Rule Number	IF	THEN	Rule Firing * Priority
1	target has not been classified & target within sensor range (60 nm) & IFF on	assert target is friendly due to IFF	Very High
2	target has not been classified & target within sensor range (60 nm) & target has achieved radar lock-on	assert target is hostile due to radar lock-on	Very High
3	target has not been classified & simulation time > 10 s & target within sensor range less than the half of the time	assert target is friendly due to going out of sensor range	Very High
4	target has not been classified & target within sensor range (60 nm) & cosine(TAA) < 0	assert classify target	High
5	classifying target & residual-based ANN selected	assert cycle residual-feature ANN	Medium
6	classifying target & invariance-based ANN selected	assert cycle invariance-based ANN	Medium
7	classifying target	assert run classification knowledge base	Medium
8	classifying target & target has been asserted hostile three out of the last four times by ANN	assert ANN classification of target is hostile	Low
9	classifying target & target has been asserted friendly three out of the last times by ANN	assert ANN classification of target is friendly	Low
10	classifying target & ANN classification is hostile & knowledge base classification is not friendly	assert target is hostile	Very Low
11	classifying target & knowledge base classification is hostile & ANN classification is not friendly	assert target is hostile	Very Low
12	classifying target & knowledge base classification is unknown & ANN classification is unknown & range < 30 nm	assert target is hostile	Very Low
13	classifying target & knowledge base classification is friendly & ANN classification is friendly	assert target is friendly	Very Low

*ordering precedence for firing rules

Figure 12 Executive Knowledge Base

classification decision-making is implemented in rules 10 through 13. In these rules, comparisons are made between the results of the ANN and the classification knowledge base, and based on these comparisons overall decisions are made. A target is considered hostile if either the ANN or the knowledge base returning hostile while other is declaring anything but friendly. This logic is implemented in rules 10 and 11. A classification of hostile is also made under the special circumstance when both ANN and the knowledge base are declaring unknown and the target is within the danger region specified by a range of less than 30 nm. Rule 12 implements this logic. A classification of friendly is asserted in rule 13 when both the ANN and KB classifiers are returning friendly.

3.4.3 Implementation of MTR System

The MTR system prototype has been implemented using our commercially-available hybrid development tool NueX. The overall system is implemented in NueX via a HyperCard stack. The stack integrates the functionality of NueX with the graphical capabilities of HyperCard. NueX provides the capabilities for the neural networks and the knowledged bases. The pregenerated trajectory data files containing sensor and filter data are read and processed within the system using the HyperCard scripting language HyperTalk. HyperTalk script is also used to drive display processing.

Figure 13 shows a view of the MTR system display as it would appear on an Apple Macintosh screen during a run. As shown in Figure 13, the majority of the display area is taken up by a radar display screen. The radar screen provides a two-dimensional display of position for all targets within the sensor range of 60 nm. The display consists of concentric circles at 15 nm intervals centered at a point which is at the crosshairs center (i.e., ownship current position). Besides the graphical information presented by the radar screen display, the MTR display also provides textual information. There are three textual elements *or fields* located on the display: 1) information concerning the status of all targets is provided in the "Target Status" field; 2) the current simulation time is presented in the "TIME" field; and 3) relevant events are presented in the "EVENTS" field as they occur during the simulation run. The "Target Status" field contains information on each of the targets as to what their real classification is and what the ANN, the classification knowledge base, and the executive knowledge base have classified it. Information in the "EVENTS" field consists of when and how the MTR system came to a make classification. The final aspects of the MTR display screen include a *button* that the user can toggle to select either the residual based (Residual Net) or the invariant-based features (Features Net) ANN and the menus located at the top of the screen. The user selects which ANN to use by clicking the mouse on the circle to the left of the desired network.

An example of what the display screen looks like during a simulation run is presented in Figure 13. Beginning with the radar screen display, 3 targets, numbered 1 through 3, are shown within sensor range. The selected ANN classifier is the invariant-based features network as indicated by the "Features Net' button being highlighted. The current simulation time as shown in the "TIME" field is 6.25s. The "Target Status" field is showing that targets 1 and 2 are actual hostiles while 3 is a friendly. This field is also showing that the ANN, the classification knowledge base, and the overall MTR system have correctly classified target 1. This is also

indicated by the information currently being displayed in the "EVENT"' field.

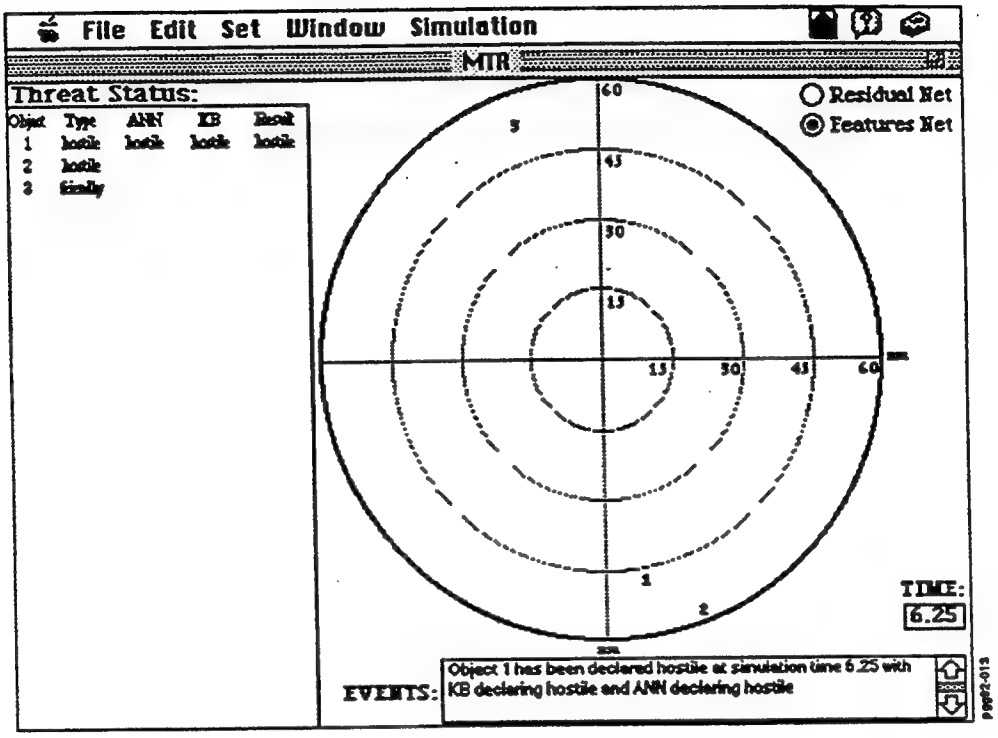


Figure 13: MTR Display interface During Simulation Run

4.0 Trade Studies

Trade studies were performed to evaluate the performance of the hybrid MTR. The goal is to demonstrate the feasibility of a such a hybrid system approach to multiple target recognition. Specific items considered include performance under varying radar sensor signal-to-noise ratios (SNR) and varying target density. For each variation, the performance of the hybrid architecture using residuals ANN vs. that using invariance ANN and the individual performance of ANN classifier vs. KB classifier vs. Hybrid Classifier are compared. Performance is judged based on three metrics: 1) probability of missed detection; 2) probability of false alarm; and 3) average detection time for hostile targets. For this study, we concern ourselves only with *initial* target classification, that is we calculate statistics concerning only the system's first target classification. Thus, all results presented pertain to how well the system performs in initially classifying targets. Once the system has classified a target, that target is no longer a candidate for classification. A

missed detection occurs if a hostile target is initially classified friendly or no classification is made (i.e., unknown classification). A false alarm is generated when a friendly is classified as a hostile. The average detection time for hostiles is the average time of detection over all the hostiles classified correctly. This statistic provides an indication of how quickly the system performs. A nominal set of parameters were used to generate these trajectories and two assumptions made in these trajectories were that ownship is stationary and target trajectories lie in the same plane as ownship, i.e., we constrain ourselves to the two-dimensional MTR problem.

4.1 Impact of Radar Signal-to-Noise Ratio Variation

Hybrid MTR system performance under varying radar sensor noise levels are presented in Figures 14 and 15 for a range of radar sensor signal-to-ratios starting from low noise levels (40 db) to high noise levels (10 db). Results for each SNR value are averages over three simulation runs generated with trajectory files having different initial conditions. To offset initial transients in sensor and filter data, delays were introduced into the simulation runs. For the invariance-based features, the first two time ticks were ignored. For the residual-based features, the first 21 time ticks (or the first five seconds of simulation time using a simulation time step of 0.25 s) were ignored to allow the residuals to ramp up and thus to improve false alarm performance. Figure 14 presents the missed detection and false alarm performance for the hybrid system using the invariance-based ANN. Comparable results for the residual-based ANN are shown in Figure 15. Included in these Figures is not only the overall hybrid system performance (as denoted by MTR), but performance of the neural network (denoted by ANN) and the classification knowledge base (denoted by KB) are also presented for comparison.

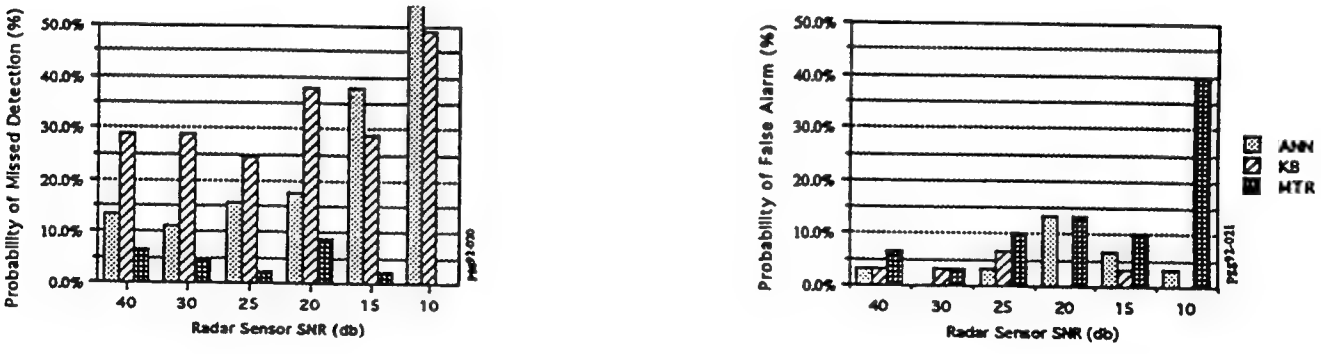


Figure 14: Effect of SNR Variation for *Invariance-Based* Features

Figure 15: Effect of SNR Variation for *Residual-Based* Features

As shown in Figure 14, the hybrid system using the invariance-based network performs well with missed detections usually well under 10% and false alarms around 10% over the range of radar sensor SNR. The only exception is the highest noise case (10 db) where false alarms are 40%. The missed detection performance of hybrid system consistently surpasses that of the individual ANN and KB classifiers. False alarm performance of MTR will always be only as good as the worst of the individual classifiers based on the logic implemented in the executive knowledge base. The reason for the very high system false alarm rate for the 10 db case is due to high range noise making targets appear much closer and thus the executive knowledge base asserts a hostile classification.

The MTR system using the residual-based features achieved very good hostile target classification performance as shown by the missed detection probabilities in Figure 15. However, the false alarms for the hybrid system are high with values around 20% and as high as 40%. The classification knowledge base is seen to perform comparably with both sets of ANN features. The cause for these false alarms is due to the performance of the residual-based feature network. The problem lies not with ANN itself, but with the hypothesis being made in the implementation. That is, the hypothesis that all targets are using the turn-to-heading intercept. In actuality, friendlies are not maneuvering and thus have zero acceleration

The hybrid system average hostile detection time performance for both sets of features for varying SNR is shown in Figure 16. The invariance-based system outperforms the residual-based across the variation in SNR. The difference usually being by a factor of three. The main reason for the enhanced performance for the invariance-based system is the delay required by the residual-based system to allow the residuals to ramp up.

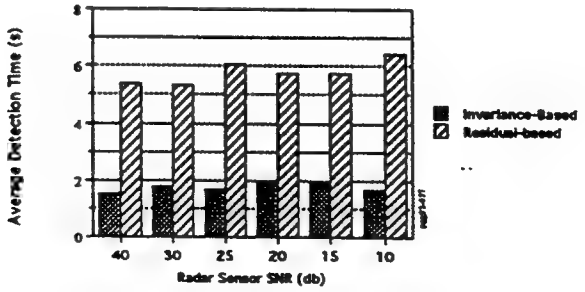


Figure 16: Effect of SNR Variation on Average Detection Time

Similar performance levels for probability of missed detection, probability of false alarm, and average detection times were achieved with varying radar sensor angular noise levels (i.e., noise levels associated with azimuth and azimuth rate measurements) and are presented in Gonsalves and Caglayan (1992). Results were generated for noise levels in the range of one-tenth (.Ix) to ten times (10x) the nominal levels of 1 deg. and 0.1 deg./s RMS. The invariance-based features performed well across the range of noise variation with respect to missed detections (around 5%) and false alarms (within 10%). Residual-based features system achieved a low level of missed detections but with a high false alarm rate (greater than 20%). Average detection times for the invariance-based system were lower than the residual-based by a factor of three-to-one. We again saw the overall hybrid system outperforming the individual ANN and KB classifiers.

4.2 Target Density Variation

The effect of an increased target environment or target density on the hybrid system classification performance is now discussed. Target density can directly impinge on system performance. Modeling of increased target density can be accomplished by increasing sensor noise. This, however has been investigated in the previous section. We implement an increased target density level by making the assumption that the MTR system can only process a limited number of targets per computation frame. This assumption is useful to account for real-time sensor and processing capabilities as well as sensor drop offs. Results are presented for target densities of 25 (nominal case), 50, 75, and 100.

Figures 17 through 18 detail the simulation performance results for invariance-based and the residual-based features, respectively. As shown in Figure 17, the missed detection performance of the invariance-based features system remains consistent across target density. However, the ANN missed detection performance deteriorates at the higher density levels, in part due to the ANN using initial data during which hostiles are still maneuvering (i.e., they may be turning about to face ownship). This indicates that the addition of other features such as the maneuver detector used in the KB classifier can improve the performance of the ANN classifier. MTR system false alarm performance is also generally consistent across the variation. For the residual-based features, no trends are noticeable for missed detection as shown in Figure 18. False performance is seen to improve at the higher density levels with false alarm probability dropping about 15% from the 50 target case to the 75 and 100 target cases. This decrease is due in part to the added delay allowing the residuals to ramp up and thus decrease the number of friendlies classified as hostiles by the ANN. Finally, the average detection times for hostile classifications across the target density variation for both sets of features are shown in Figure 19. Again, the invariance-based outperforms the residual-based features across the variation by over a factor of three at the lower density levels to a factor of 2 at the high density levels. Both features demonstrate increasing detection times with increasing target density due to the added delays.

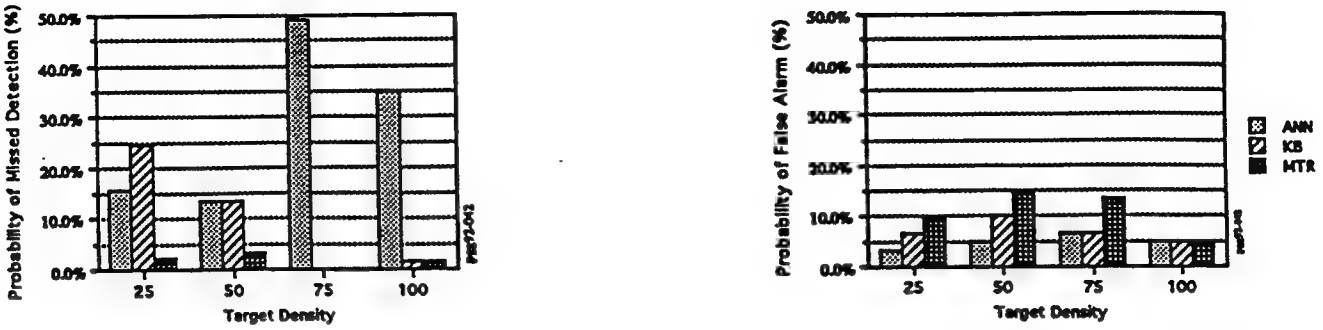


Figure 17: Effect of Target Density Variation for *Invariance*-Based Features

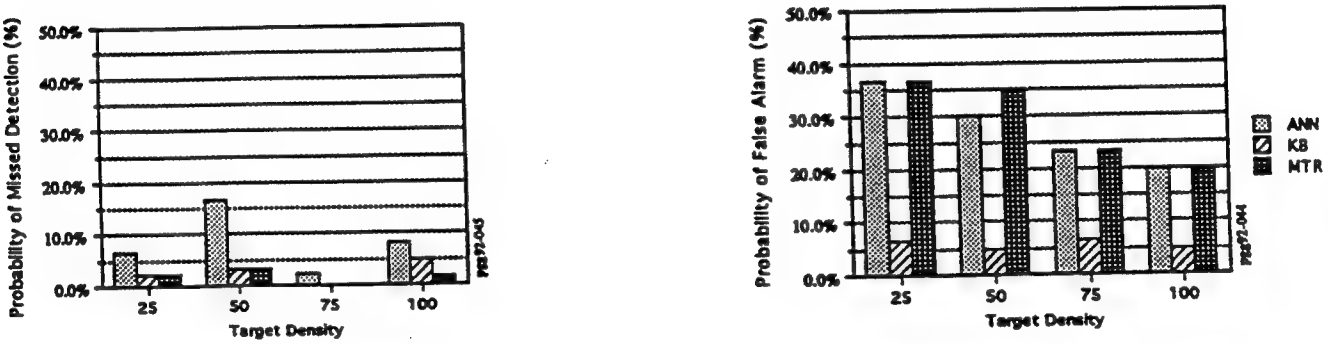


Figure 18: Effect of Target Density Variation for *Residual*-Based Features

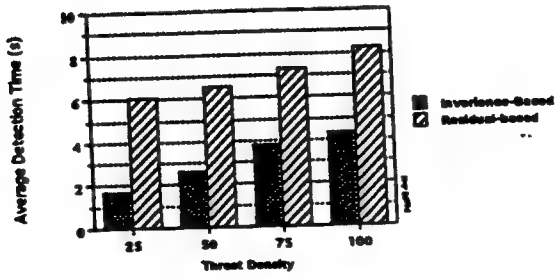


Figure 19: Effect of Target Density Variation on Average Detection Time

5.0 Summary and Conclusions

We have specified, designed, and implemented a hybrid MTR system architecture incorporating conventional signal processing and probabilistic tracking algorithms with ANN and KB classifiers, and a KB decision maker. Two competing hybrid MTR systems have been implemented using the developed architecture: one with invariance-based ANN input features and another with ANN features generated by a conventional tracking estimator. In general, our demonstration study shows the following that a hybrid neural network expert system approach to NM incorporating conventional signal processing and tracking estimators is feasible. The performance of the hybrid classifier is higher than the performance of the individual ANN and KB classifiers comprising the hybrid system. The invariance-based sensor feature driven ANN outperforms the tracking estimator residuals driven ANN.

6.0 Acknowledgement

This work was sponsored by the Naval Surface Warfare Center, Silver Spring, MD under contract No. N60921-91-C-0186. The authors thank the Technical Monitor, Dr. Ali Farsaie, for his support and assistance.

7.0 References

Anderson, J.A., Rossen, M.L., Viscuso, S.R., et al. 1990. "Experiments with Representation in Neural Networks: Object Motion, Speech, and Arithmetic." *Springer Series in Synergetics,* Vol. 45: pp. 54-69.

Bar-Shalom, Y. 1978. "Tracking Methods in a Multitarget Environment."IEEE *Trans, on Auto. Control,* Vol. AC-23, No. 4.

Bamard, E. and Casasent, D. 1991. "Invariance and Neural Nets." *IEEE Transactions on Neural Networks*, Vol. 2, No. 5.

Caglayan, A.K. and Allen, S.M. 1990. "A Neural Net Approach to Space Vehicle Guidance." *American Control Conference*. San Diego, CA.

Chang, C.B. and Tabaczynski, J.A. 1984. "Applications of State Estimation to Target Tracking." *IEEE Trans. on Auto Contr.,* Vol. AC-29, No. 2.

Gonsalves, P.G. and Caglayan, A.K., 1988. "A Hybrid Neural Network Expert System Approach to Multiple Target Recognition." Report No. R9112 1, Charles River Analytics, Cambridge, MA.

Kuczewski, R. 1989. *TRW Investigation of Scan-Scan Correlation and Multi-target Tracking.* DARPA Neural Network Study, Lincoln Lab Report No. 840.

Lin, C.E. and Chen, K.L. 1991. "Automated Air Defense System Using Knowledge-Based System." *IEEE Transactions on Aerospace and Electronic Systems, Vol. 27, No. 1.*

Lorczak, P.R., Caglayan, A.K. and Eckhardt, D.E. 1989. "A Theoretical Investigation of Generalized Voters for Redundant Systems." *FTCS 19.* Chicago, IL.

Roth, M.W. 1990. "Survey of Neural Network Technology for Automatic Target Recognition." *IEEE Trans. on Neural Networks*, Vol. 1, No. 1.

Artificial Neural Networks are Indeed Smart, Biology-Inspired Research (SBIR) Tools[1*]

Srinivasan Raghavan and Laveen Kanal[2†]
LNK Corporation Inc.
6811 Kenilworth Ave., Suite 306
Riverdale, MD 20737
(301) 927 3223

Abstract: Artificial neural networks (ANNs), as a next generation technology, hold the potential to solve a number of complex problems. Major advantages of ANNs include their capability to combine adaptation with parallel processing, their distributed computing nature, reliability due to distributed representation, and flexibility due to collective decision making. Inspired by biological neurons, this technology lends itself to a fine-grained parallel hardware implementation for real-time needs. The DOD has been a key player in promoting the application of neural networks for a number of areas including target detection and discrimination, tracking, acoustic signal processing, speech processing, radar signal processing, and image processing. LNK has been involved in applying neural networks for various applications in signal and image processing under various Navy SBIR efforts ranging from one dimensional Radar Cross Sections analysis to Automatic Target Recognition from optical imagery. In addition, LNK has also developed integrated systems combining neural networks and expert system for a sensor fusion application of Automatic Feature Extraction from imagery. A number of innovative concepts have been developed at LNK including new learning rules, hybrid neural networks, dynamic trimming of neural networks, and network visualization. In this paper, we present a short summary of each SBIR effort and the related accomplishments at LNK.

1.0 Introduction

Artificial Neural Networks or simply neural networks, have a relatively long history of research beginning with the early part of this century. Only in the recent past, have they found increasing use in a number of applications involving intelligent pattern recognition and classification, prediction, retrospective interpolation of data, and in finding data regularities and irregularities. Such a growth of interest in using neural networks can be attributed to several advantages that

---

[1*] The work described in this paper has been performed under the following Navy SBIR contracts N62269-90-C-0567, N60921-89-C-0030, N0024-90-C-3814, and N00014-89-C-0011. The authors would like to thank Naresh Gupta, Barbara Lambird, David Lavine, Gretchen Bailey and Jason Meyer for their work on these SBIR projects.

[2†] Also, Professor, Department of Computer Science, University of Maryland, College Park, MD.

neural networks offer: adaptability, easily trainable nature, flexibility, absence of any need for expert knowledge, and self-programming ability. The remarkable growth in research, as well as commercial simulators, have accelerated the use of neural networks in other areas. LNK has been involved in the research and use of neural networks for several applications over the past 7 years. Since the renewed interest in neural networks has gained considerable momentum, neural networks have tremendous potential for solving a variety of challenging problems associated with data understanding.

Neural networks are structures consisting of a large number of simple computational units with variable interconnection weights between them. The computational units, also known as "neurons", are capable of elementary operations such as addition and thresholding. Neural networks "adapt" to the data behavior by adjusting or self-programming the weights using a training procedure. Two major classes of training methodologies exist: unsupervised learning and supervised learning. Networks based on unsupervised learning divide the pattern space into bins of similar patterns in a self-organizing fashion. Networks based on supervised learning divide the pattern space into classes specified by the input-output example pairs in the training data set. Examples of unsupervised neural net classifiers include Kohonen's network and Adaptive Resonance Theory. Examples for supervised neural networks include feed forward Back Propagation networks and Perceptrons. In addition to the two major types of learning used in timeless events, other related concepts such as spatio-temporal learning provide the capability to learn time-related behaviors of data patterns. The objective of this paper is not to provide a grand treatise on neural networks, but rather to present an overview of the applications in which we can show that neural networks can outperform other competing technologies. We discover the weeknesses of neural networks and complement them through proposing hybrid systems.

The reminder of the paper is organized as follows. We begin with an overview of Navy SBIR work at LNK in the next section to provide a quick glance to the reader. In section 3, our experimentation with neural networks on locating the sonar source for reduction of submarine noise. A multi-level hybrid neural network architecture for target recognition from Radar Cross Sections is presented in Section section 4. We show in section 5 that neural networks when combined with expert systems result in a robust classification of terrain imagery for automated feature extraction purposes. Section 6 discusses our work on a multiple target tracking algorithm which can be implemented in symmetric and asymmetric mean-field thermodynamic neural networks [Hellstrom and Kanal 1992]. Section 7 presents concluding remarks.

2.0 Overview

The following are the four recent Navy SBIR neural network technology projects completed at LNK .

For the Naval Sea Systems Command, LNK developed an ANN based technique to estimate location of vibration sources on board submarines. This technique makes use of acoustic transients measured by sensors in the same room as well as adjacent rooms with partially reflecting walls. The arrival times of the signals are the primary features used by the neural networks. The accuracy of source localization ranged from 4% to 9% of the room dimension for

various levels of noise and reflection conditions.

Conventional ANN paradigms often suffer from significant disadvantages when employed individually for a specific application. A hybrid structure can evolve by combining various paradigms for the benefit of exploiting their complementary advantages [Kanal and Raghavan 1992]. LNK has designed a hybrid neural network structure for Naval Surface Weapon Center during a Phase II SBIR effort to analyze Radar Cross Sections (RCS). The resulting performance improves from a mediocre recognition rate of around 65% to 95-98% for target recognition from RCS.

In addition to combining networks, LNK believes that optimal solutions to many problems can be obtained by integrating technologies as well. Expert systems, fuzzy logic, and genetic algorithms are examples of such powerful technologies whose advantages can be exploited through developing an integrated system. For Naval Air Development Center, LNK has combined neural networks and expert systems to achieve a common platform to integrate symbolic feature databases and sensors for terrain feature extraction.

Neural networks offer a convenient means to achieve optimal solutions for the target motion estimation problem. LNK, under an SDIO - Office of Naval Research Phase II SBIR project, has built a novel passive imagery-based multiple target tracking algorithm capable of tracking objects under clutter, camouflage, and occlusion. This tracking algorithm yields itself to a thermodynamic neural network with mean-field annealing.

These projects are described in separate sections to facilitate a quick reference to the specific topics in which a reader may be interested in.

3.0 Sonar Source Location in Submarines Using Neural Networks

Silencing of our own submarines is essential, since their ability to effectively perform missions frequently depends on their remaining undetected. The silencing problem can be decomposed into two parts: (1) location of sound or vibrational sources, and (2) quieting of the sources. In this section, we describe the research effort supported by NavSea to address the problem of estimating the location of vibration sources. Throughout this section, we use the term source to refer to a structure on a submarine producing vibrations which can be transmitted along the structural members of a ship and be detected by an accelerometer.

Many factors contribute to the complexity of estimating source locations on submarines. Sources may take a variety of forms, such as bearings in a motor. The characteristics of the signals produced by sources are not often known in advance, and the characteristics of a source may vary over time as the parts wear out. The vibrations emanating from a source will travel in multiple directions and the resulting waves will cross many times, sometimes reinforcing and at other times attenuating the resulting waves. Waves can be reflected or partially transmitted at boundaries between different materials, resulting in complex waveforms. Waves can also experience frequency dependent distortion, further complicating the structure of the resulting waveforms.

Two approaches to the problem of source location estimation are plausible. In the first approach, detailed structural modelling of signals and acoustic characteristics of submarine structures are developed and one can try to predict the signals which might be received at one or more sensors, for a given type of source. The information required to develop such models may not be available and the numerical problems in such an analysis may be overwhelming because of the large numbers of variables involved.

The second approach is a more practical approach to source location estimation. We assumed that a source is available which can be moved around a submarine, and sensors are available to detect the signals resulting from the source at each of its locations. This procedure provides us with a set of source locations and sensor responses. Our approach is to consider this data as a set of samples of a function which takes a set of sensor responses and yields a source location. We then use a function approximation technique based on neural networks, to estimate the function at values other than the samples. The neural network is trained on the samples and the result is a function, which when fed with a set of sensor responses as input yields a source location. As the complete signal at a sensor resulting from a source is a complex entity, this signal is replaced by a small set of extracted features, such as wave arrival times, before it is fed into the neural network. Thus, in this application the neural networks alternatively play the role of a function approximator and a predictor.

3.1 Data Generation

For lack of adequate real data for testing as well as training the proof-of-concept system, we chose to generate synthetic data from a known submarine model and its material properties. The submarine model that we used is a two dimensional simplification of the true structure of the submarine. The submarine is represented as a set of paths along which vibration can be transmitted, and the path intersections. We chose to use a two dimensional submarine to facilitate the generation of simulated data, to keep the physics of the situation much simpler. Our model of the submarine, is given in terms of one dimensional transmitting paths. Each path attenuates vibrations exponentially with length travelled. To facilitate computation, we have assumed that this attenuation as well as the speed of propagation of vibrations is independent of frequency. Where the paths meet, it is assumed that there is either a branching of paths and/or a change of materials. At such junctions, the vibration's amplitude will split (possibly unequally) among the available options. The typical profile of the vibration source is viewed as a damped harmonic oscillator and thus has the form of a decaying exponential multiplied by a cosine function.

With the specifications of nature of materials, structure of the submarine, and locations of sensor, a simple code has been written to analyze the tree path of propagation and record the peak amplitude and time of arrival at every sensor location. (see Lavine et. al [1990] for details) The attenuation is proportional to the distance as mentioned earlier. We find that for proof-of-concept, this simple way of path traversal for computing the time of arrival and using exponential decay model to estimate the signal amplitude is adequate.

3.2 Neural Network Based Prediction

The primary goal of this research effort was to determine the accuracy to which the location of a vibration source can be located using a neural network. We employed a feedforward backpropagation (BP) neural network to locate the sonar sources by transforming the problem as a functional prediction problem. The choice of a BP network here is most appropriate because of its universal mapping and functional approximation abilities. A typical BP network, as shown in Figure 3.1, consists of three or more layers of neurons of which the first layer signifies input, the last layer signifies output and the layers in between the first and last layers are known as hidden layers. These networks are feedforward in the sense that during classification input information can only be propagated forward from input through output layers. However, the training algorithm used to train these networks allows the error between expected output and the actual output for a training sample to propagate backward from the output layer to the input layer through the hidden layers and hence such nets are called as feedforward BP networks or simply BP networks.

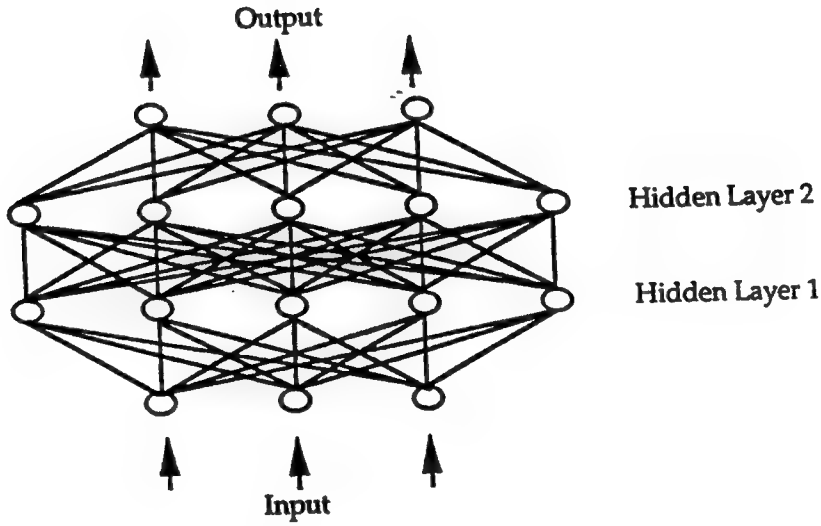


Figure 3.1 Feedforward Backpropagation Network

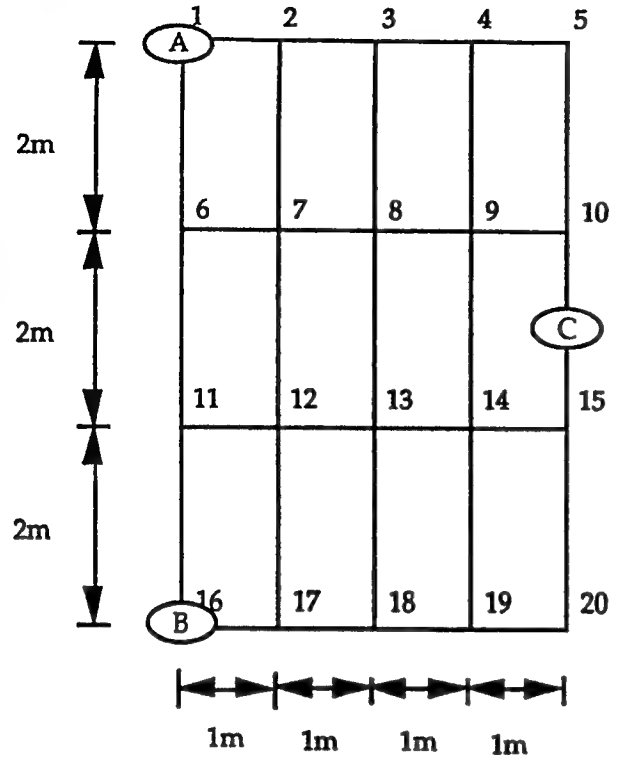


Figure 3.2a: Three sensors A,B, and C are distributed in a 4X6 room. There are totally 20 nodes and the vibration paths in between the nodes are represented by lines.

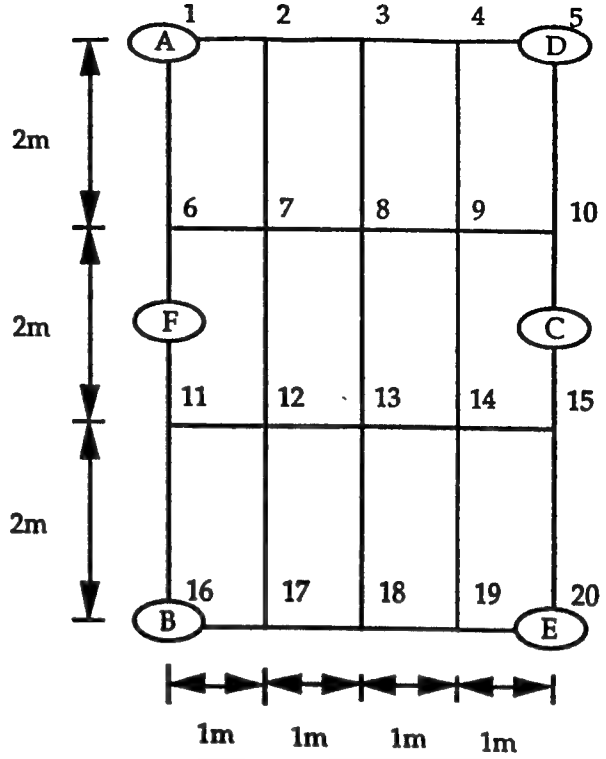


Figure 3.2b: Six sensors A,B, C, D, E, and F are distributed in a 4X6 room. This nonminimal sensor configration provides additional robustness.

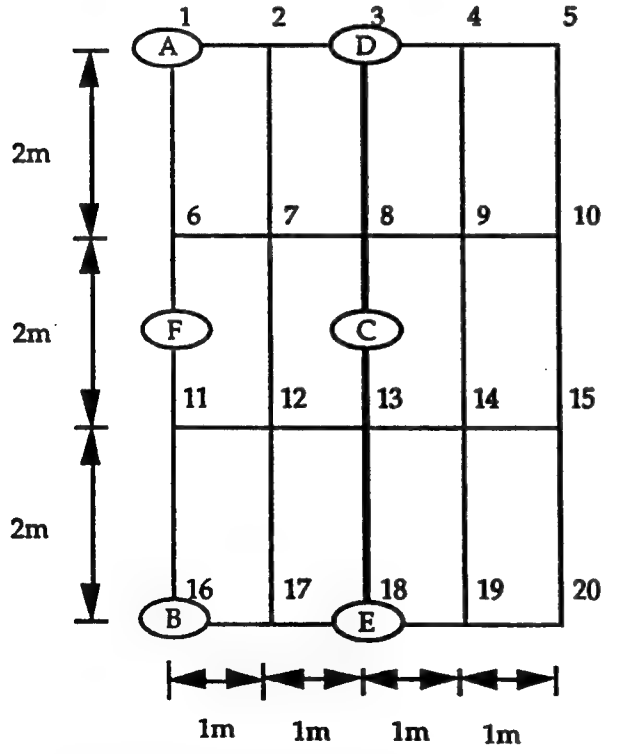


Figure 3.2c: Six sensors A,B, C, D, E, and F are distributed in one of the two 2X6 rooms. The wall between node 3 and 18 is not reflective.

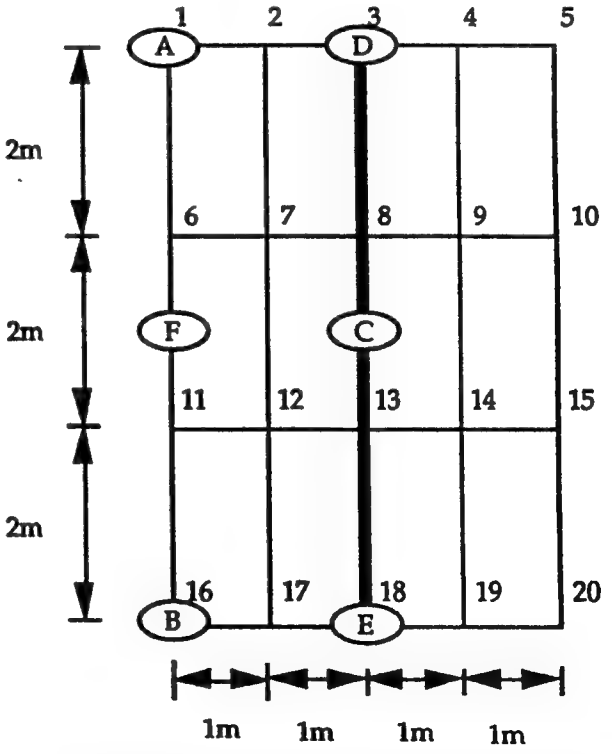


Figure 3.2d: Six sensors A,B, C, D, E, and F are distributed in one of the two 2X6 rooms. The wall between node 3 and 18 is reflective.

		Training Set	Test Sets			
		Noise (μg^2)	Noise (μg^2)			
		$5x10^6$	0	$1x10^6$	$5x10^6$	$1x10^7$
Total	Number of Points	153	40	40	40	40
	μ (cm)	16.2	30.8	33.7	43.5	57.2
	σ (cm)	11.4	43.8	44.2	56.8	67.0
	σ_μ(cm)	0.9	6.9	7.0	9.0	10.6
Excluding 3σ Outliers	Number of Outliers	2	1	1	1	1
	Median (cm)	12.8	17.1	19.1	20.9	22.8
	μ (cm)	15.5	24.8	27.9	37.9	51.6
	σ (cm)	9.8	22.9	25.5	45.4	57.8
	σ_μ(cm)	0.8	3.7	4.1	7.3	9.3

Table 3.1 Location Errors for Two Rooms with Reflection Model Explained in Experiment Set #4.

For details on the BP learning algorithm, see [Rumelhart et. al. 1986]. Here we comment on experimentation. The training data was generated by selecting a source location and a noise level, propagating the signals from this source location through the network, computing the resulting signal at each sensor, and computing features for the signal arriving at each sensor. In each experiment, this procedure was repeated with 153 source locations to generate the training data. The sound sources were equally spaced at 25 cm intervals around the grid. Test data for each experiment was obtained by randomly placing a sound source on the grid, determining the resulting signal at each sensor, adding noise to this signal and extracting the features from this signal. This process was repeated for a total of 40 random source locations.

3.3 Experimentation

Four experiments were done in each experiment set, each experiment corresponding to a different noise level. Two types of features were estimated from the signals: (1) the first time of arrival of a signal at a sensor, and (2) the time at which a smoothed version of the derivative of the signal was maximized. The algorithm performed more reliably with the second feature, while the first arrival time has the advantage that it is easier to analyze its performance theoretically. We provide detailed results only on Experiment #4 since it represents the maximal complexity.

3.3.1 Experiment Set #1 - First Arrival Times

In this set of experiments, the arrival time at which a signal first arrives at each sensor was estimated. Two groups of experiments were done in experiment set #1. In the first group, three sensors were used and were placed as shown in Figure 3.2a. The results with this configuration were not encouraging. The problem is that three sensors are, in general, required to uniquely locate a source and it is not always possible to reliably extract the first arrival times at all three sensors for a given source. Arrival time estimation failure at one sensor is enough to cause a bad location of estimation. The two primary problems in Experiment Set #1 were the difficulty in reliably extracting first arrival times and the lack of additional sensors to provide redundancy for greater estimation robustness in the presence of noise. Both of these shortcomings were addressed in Experiment Set #2.

3.3.2 Experiment Set #2 - Maximal Derivative, One Room

In this set of experiments, a one room configuration was used with six sensors placed as shown in Figure 3.2b. At each sensor, the incoming signal was processed to estimate the time at which the derivative of the signal is maximum. This feature was selected since it could be more reliably extracted than the first arrival. It should be noted that even if the first arrival times are known exactly, the location of a source cannot be exactly determined if the source is outside the smallest rectangle bounding the sensors. With zero noise, the mean error was 0 .22 m or approximately 4% of the maximum room dimension. Ninety percent of the errors were half a meter or less. When testing the system at a noise level at which the system was trained (5.0×10^6 mg^2), the mean error rose to 0.29m.

3.3.3 Experiment Set #3 - Maximal Derivative, Two Rooms, Non-Reflecting Wall

In order to achieve more realistic results, we decided to consider a more complex problem than that considered in Experiment Sets #1 and #2. In Experiment Set #3, we subdivided the structure into two equal size parts of size 6m x 2m, as illustrated in Figure 3.2c. We considered the resulting structure to represent two rooms with all sensors being contained in the room to the left. In order to simplify the problem, in this experiment set, we assumed no reflectivity at the wall between the two rooms. This restriction is lifted in Experiment Set #4. This experiment is equivalent to a single large room with the sensors located so as not to surround the room. With zero noise, the mean error was 0.22 m or approximately 4% of the maximum room dimension. 90% of the errors where 0.6m or less. When testing the system at the noise level at which the system was trained, 5.0×10^6 mg^2, the mean error rose to 0.3m. The errors in this experiment were similar to those of the single room case.

3.3.4 Experiment Set #4 - Maximal Derivative, Two Rooms, Reflecting Wall

In this experiment we have two rooms, as in Experiment Set #3, but in this experiment, the wall is partially reflective with a reflectivity coefficient of 0.8. This configuration is shown in Figure 3.2d. The results of this experiment are shown in Table 3.1. With zero noise, the mean error was 0.34m or approximately 6% of the maximum room dimension. 90% of the errors where half a meter or less. When testing the system at the noise level at which the system was trained 5.0×10^6 mg^2, the mean error rose to 0.43m. The presence of the reflecting barrier increased the error by nearly 50% over the same setup without a reflecting barrier.

From all these experiments it has become clear that neural networks can help achieve accurate locating of sonar sources in submarines with the advantages of inexpensive modelling, ability to tolerate noise and lack of sophistication in generation of data, and adaptability to continuously learn relationships between sophisticated structures and source locations.

4.0 Analysis of Radar Cross Sections Using Neural Networks

The radar echo characteristics of a wide variety of bodies have been studied since World War II. Radar Cross Sections (RCS), which measure the relative strengths of incident and reflected waves on a target, can be combined to form a range profile, which provides a one dimensional representation of an object, and shows considerable promise for application in target recognition. RCS refers to a measure of comparison of the power density of the reflected wave at the radar site with that of the incident wave at the target. The RCS range profile gives the strength of the radar return as a function of distance from the receiver. By developing a correspondence between the distances at which strong returns have occurred and the locations of strong scatterers on various targets, it is often possible to classify a target.

Due to its obviously sensitive nature, it is hard to find earlier work reported in the literature on recognition of RCS range profile. By ignoring the additional complexities associated with RCS, we can treat RCS range profile merely as a one dimensional signal. There is considerable

interest in using neural networks for identifying specific characteristics of one dimensional signals in a number of applications. Examples of these applications include speech processing [Lang et. al. 1990] and acoustic signal processing [Gorman and Sejnowski 1988]. In all these applications, we find that the researchers are more interested in applying only one of the popular neural network paradigms for the chosen application and evaluating the performance. Yet, different networks are based on different fundamental principles, so it should be expected that some networks perform better than others in alleviating specific complexities in the problem. Thus, a better approach is to employ multiple networks and devise an integration framework to fuse the decisions from the networks. This section discusses our SBIR Phase II work with Naval Surface Warfare Center (NSWC) to classify RCS signals of target aircraft using a hybrid neural network.

Hybrid networks proposed in the literature mostly involve hybrid combinations of learning theories. Better termed as hybrid learning-based networks, these networks exploit the complementary advantages of different types of learning in a single structure. For instance, the counter propagation network proposed by Hecht-Nielsen [1988] integrates Kohonen and Grossberg learning in a single five-layered network to achieve optimal performance on self-categorization. We believe that it is also profitable to couple networks on a structural basis so that each class of networks helps solve a sub-problem. We will refer to this class of hybrid networks as structurally hybrid networks. We have earlier proposed a structurally hybrid architecture of unsupervised Adaptive Resonance Theory I networks and supervised feedforward backpropagation networks to achieve object recognition from imagery [Raghavan et. al. 1991]. While, the unsupervised networks in this architecture extract geometric features without explicit enumeration as in the case of Neocognitron [Fukushima 1988], supervised networks help interpret the geometric features to form object shapes. Based on this experience, we find that by coupling multiple networks which are appropriately suited to solve one or more sub-problems, we can achieve an optimal performance in a multi-level classifier.

4.1 Radar Cross Sections

RCS of an object can be thought of as the total cross sectional area of an imaginary ideally conducting sphere which scatters the same amount of power as the object of interest will do in a given direction for a given incident field [Ruck et. al 1970]. Various factors that influence the range profile received at the radar are:

- transmitting system,
- Propagating medium,
- RCS of the target,
- propagating medium,
- receiving system, and
- polarization effects.

By varying the wavelength, different values of the RCS are obtained. Stepping the radar cross section through a whole range of wavelengths (or, equivalently, frequencies), a function of RCS versus frequency is obtained. By applying an inverse Fourier transform to this function, we get

a function of RCS versus range, which is called a RCS range profile or simply RCS. Henceforth, the terms "RCS range profile" and "RCS" will be assumed interchangeable in this paper and both will allude to the same meaning.

The RCS has several desirable characteristics. First, changes in the range profile due to obscuration of a part of an aircraft are localized in the range profile, while they are not localized in the frequency domain. This localization ensures that parts of the signal remain almost unchanged and can serve as a reliable basis for classification. Second, minor changes to the aircraft models, such as the addition of missiles, are localized and should not greatly affect recognition. Finally, the range profile admits a partial interpretation in terms of object shapes, which can be comprehended by humans, making this representation appealing to design an Automatic Target Recognition (ATR) system.

In our work, the RCS data for the target aircraft was generated using a commercial simulator. The need to simulate RCS data stems from two major reasons. First, there is a constant desire to minimize RCS of military objects during their design phase, in which case simulation is the only way to measure RCS of the "unbuilt" object. Second, nonavailability of real world RCS data for many applications forces us to explore accurate and efficient methods to simulate RCS. Generation of RCS is a complex problem since the strength of return from the target depends on several factors including shape of the aircraft, radar look-angle, material, secondary reflections, and other medium-related factors. A simulator which takes into account all these factors is expensive because modelling every one of these factors is a complex task. In addition, the physical mechanisms underlying some of these factors are not completely understood, and the resulting simulations are not entirely realistic. However, many of these factors may not have a significant influence on the RCS pattern.

We made use of a commercial simulator, known as Denmar Back Scatter Method (DBSM), to simulate RCS for target aircraft. The DBSM radar simulation package captures the two most important aspects of RCS simulation namely the object geometric shape and radar look angle. Substantial variations and complexities to RCS of the aircraft are introduced even when these two major factors alone are included. Thus, this may be viewed as a first-order approximation to a realistic scenario to design and test the concept of a robust classifier. In addition, since we only had aircraft drawings as a basis for our models, and could not model the detailed curvature of the aircraft surfaces, a component modelling system was much easier to use. DBSM has facilities for forming models by specifying model components in terms of simple shapes such as plates, cones, cylinders, and frustra. Once the model is formed, the system generates RCS as a function of frequency for a given azimuth and elevation viewing angle. DBSM uses different RCS computation routines tailored to each of the simple shapes, and then combines the RCS from each of these to form a composite RCS. An inverse Fourier transform is computed to yield RCS as a function of range for the fixed azimuth and elevation viewing angle. The resulting data is called a range profile.

4.2 RCS Classification Using a Multi-Level Hybrid Neural Network Structure

A cursory look at the problem of classifying RCS patterns suggests that the myriad variations in

RCS for the same object under different views appear to pose formidable challenges in classification. Even under simplistic conditions of ignoring secondary reflections, RCS for complex objects change considerably from one radar look angle of azimuth and elevation to the immediately next angle. Fortunately, large changes in RCS seem to occur around a fewer number of look angles than continuously for change in every degree of the azimuth and elevation. A crude example to demonstrate this behavior is the appearance of objects under different viewing conditions. Features of objects such as their edges and faces appear to be constant over a large part of rotation, yet change dramatically at some part of rotation due to self-occlusion. A similar behavior is noticed in RCS analysis as well.

We first identify the issues that remain to be addressed in classifying RCS patterns of target aircraft and later choose the neural network paradigms which help address these issues.

- The foremost issue in RCS classification is the variability of RCS signatures as the angle between radar and aircraft changes. The variations in RCS are significant when obscuration of aircraft components occurs when the aircraft turns away from the radar. We will refer to this problem as the *obscuration* problem. Absence of RCS peaks is often noticed when the target is occluded. The classifier is, therefore, required to possess generalization capabilities to recognize the target class under a wide range of look angles.

- The second issue is the *noise* problem. Typical noise levels in the radar receiver are in the range of -40dB to -30dB. It is often found that the peak return of some targets barely exceeds -25dB in our simulations and thus noise tends to cause enormous difficulty in identifying features of the RCS pattern exclusive to the specific target. The classifier is thus required to be relatively insensitive to the noise contained in the received signal.

- The third issue is the change in RCS patterns due to inclusion/removal of additional components to/from the aircraft such as missiles and payloads. The sharp corners and edges of these components (e.g., fins and nose) introduce additional peaks to the RCS patterns due to the strong return from these components. We will refer to this problem simply as the *payload variation* problem. The classifier should be able to "absorb" the additional changes due to payload variations.

- The fourth issue is the expected local shifting in the RCS pattern due to variations in target range relative to the radar. Since RCS range profile signifies strength of the return measured at every quantized range point, the target range is likely to "shift" the RCS pattern along the range axis. Improper localization of the radar return will amount to local shifting, referred to as the *shifting* problem, and thus calls for shift invariance in the classifier.

- Finally, in the current application, in addition to classifying the targets of the designated set, the classifier is also required to distinguish between target and "nontargets". This requirement, known as *nontarget identification*, places enormous restrictions on the choice of classifiers since it means that the classifier should not simply select the

"closest" prototype to the given RCS pattern.

In addressing these complexities, a key observation to motivate our design of a multi-level classifier is that different neural network paradigms involve different principles of learning which help solve different types of complexities in the RCS data. For instance, the feedforward backpropagation (BP) networks achieve classification based on learning to minimize the mean-squared error between input-output pairs in the example set. This particular nature of feedforward networks helps generalization of the learned mapping between the RCS and aircraft classes which, in turn, addresses the issue of dealing with arbitrary poses with only a small training set. However, these networks are poor in discriminating nontargets which have not been learnt, resulting in a higher false alarm rate. Therefore, our goal is to combine various paradigms for their complementary advantages in such a fashion as to address all the complexities anticipated in the recognition of RCS.

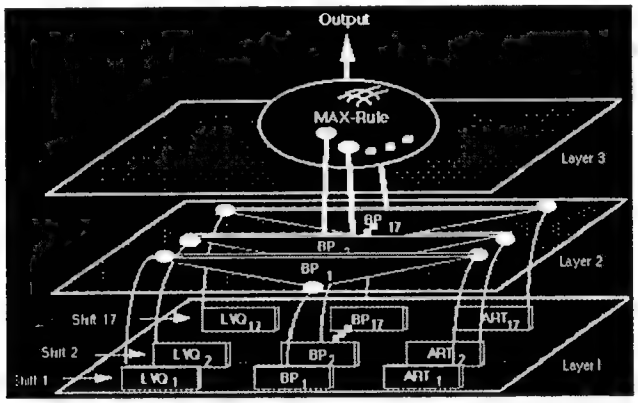


Figure 4.1: A Multi-Level Hybrid Neural Network RCS Classifier

The proof-of-concept system is a three level classifier structure as shown in Figure 4.1., in which the first two levels involve an overlapped configuration of neural networks and the third level employs a simple majority rule mechanism. The first level of the structure consists of a combination of Adaptive Resonance Theory II (ART) networks [Carpenter and Grossberg 1987], Learning Vector Quantization (LVQ) networks [Kohonen 1988], and Backpropagation (BP) [Rumelhart et. al. 1986] networks, each of which receives the preprocessed RCS input independently. Some salient features of our system at the first level are:

- The networks are arranged in an overlapped fashion to handle global shift on the range axis.

- BP networks provide the generalization capability to handle the obscuration problem.

- Because LVQ networks achieve classification based on the near-neighborhood learning principle.they seem to handle the local shifting problem.

- ART networks, due to their stable learning property, help reduce the false alarm rate by rejecting nontargets as invalid signals.

The second level of the system incorporates another spatially overlapped population of BP networks. Each network at this level receives, as input, the output activations of three networks of the first level at the corresponding location on the range axis. Output units of the second level BP networks are thresholded with a cut-off to enable a binary decision on the aircraft class the target belongs to. Finally, at the third level, a majority rule fires by taking the maximum number of output cells which are on at the second level as the collective decision made by the system.

	Target 1	Target 2	Target 3	Target 4	Target 5
BP	92 %	98 %	100 %	100 %	100 %
ART II	100 %	97 %	100 %	80 %	100 %
LVQ	93 %	100 %	99 %	90 %	100 %
Hybrid	100 %	100 %	100 %	99 %	100 %

Table 4.1: Sample Results of the Hybrid Network Classifier for RCS

A few sample results of testing the system are shown in Table 4.1. Particularly, noteworthy from these results is the ability of collective decision making exhibited by our hybrid system. Apart from the samples provided in this paper, in general, we could achieve a recognition rate of 90-100% with complexities including -30dB noise, added missiles, 60^0 azimuth and elevation variations and nontargets in the test set.

5.0 Sensor Fusion Using a Hybrid Expert-Neural Network System

Feature extraction from multi-source imagery and collateral sources, also known as terrain image understanding, can be of key importance in many military and civilian activities, including mission analysis, navigation, automatic target detection and recognition, spatial database updating, evaluation of environmental effects, and prediction of agricultural yield. Even though, the desired features include both natural as well as man-made features, discussion of this section is limited to locating only natural features. In order to detect these features in all weather/day conditions and improve reliability, more than one sensor can be used. Collateral data such as geographic or relational databases can be very useful. The amount of human effort required to make a decision using multiple sources can be enormous and time consuming. We find that a hybrid combination of neural networks and expert systems is a very useful to automate the process of integrating heterogeneous data sources. In this section, we describe our SBIR experience with a hybrid system to extract terrain features, known as InFuse, developed under our Naval Air Development Center (NADC) SBIR Phase I.

Neural network based fusion of multiple sources is recently becoming more popular. Rajapakse et. al. [1990] proposed a five-layered neural network architecture for object recognition. It is not clear how this architecture will learn textural features since it uses principles of geometric pattern learning similar to the Neocognitron [Fukushima 1988]. In terrain classification *a priori* known sensor behavior is very useful to allow a variable gain control mechanism in weighing the input from different sensors. For example, in Synthetic Aperture Radar (SAR) imagery water regions stand out

well consistently during all seasons due to the poor radar reflection, yet forests in SAR may not exhibit a consistent behavior due to absence of foliage in winter. Such important information can not be easily captured in neural networks.

Hybrid systems hold the key to the problem of integrating multiple heterogeneous data sources. Better performance results when hybrid systems are used to solve several complex problems [Kanal and Raghavan 1992]. This is because different computational tools are often found to possess complementary advantages. We have shown that a robust and reliable feature extraction system results when a hybrid combination of neural networks and expert systems is used to fuse multi-source imagery and spatial databases. In the current implementation of the system, SAR and optical imagery were used to extract forests, water bodies, and fields by fusing SAR and optical imagery with the aid of an expert system and a geographic database. In fusing the imagery, we observe that two types of fusion are possible. One of them, known as decision-level fusion, relates to the concept of individually classifying the imagery first followed by fusing the class decisions at the next level. The other type, known as data-level fusion, relates to the notion of treating the textural features from all the images as a single input vector towards the goal of classification. We present our observations in experimenting with the system and some empirical conclusions derived from testing.

5.1 InFuse Architecture

The architecture of InFuse, as shown in Figure 5.1, is a three layered system in which an expert system is housed at the third level and a population of neural networks is configured to work at the second level, while a library of basic image processing and data representation schemes are made available at the first level. The three levels are loosely coupled and are allowed to interact with each other.

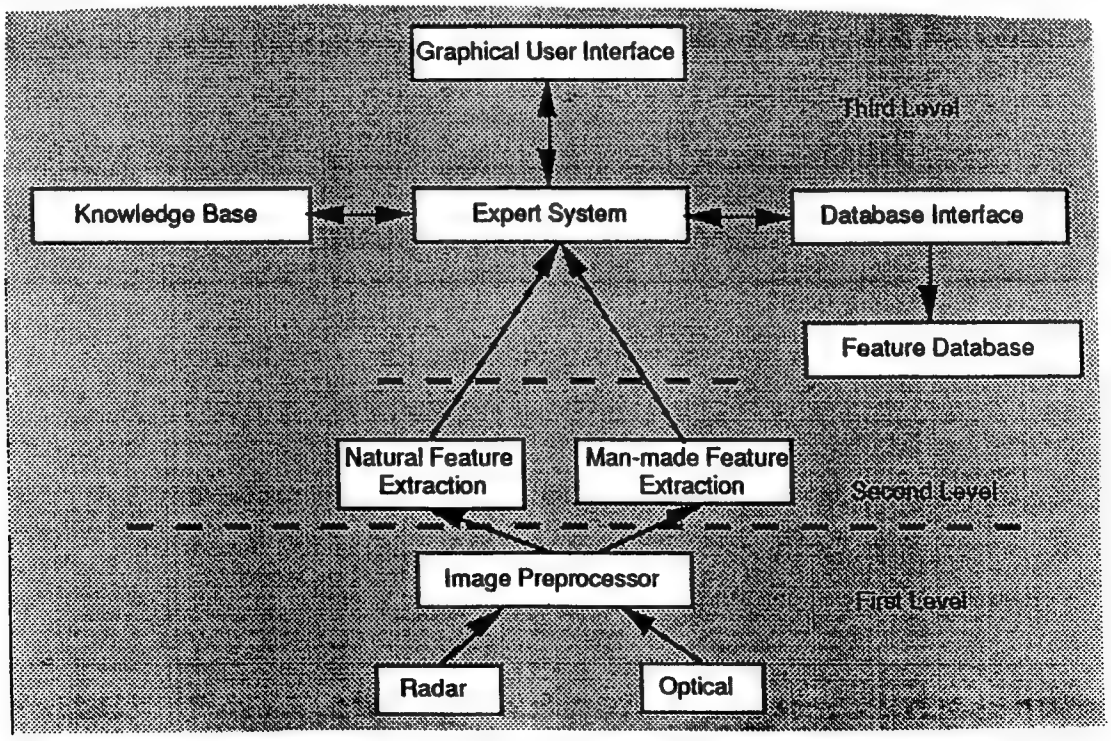


Figure 5.1: The Generic Architecture of InFuse

Beginning with the first level, the image preprocessor is equipped with a library of filtering schemes, image registration software, and image clean-up routines. The image preprocessor also contains routines to achieve discontinuity-preserved smoothing and edge detection [Blake and Zisserman 1987] which form the input to man-made feature extraction. Edge features are of little use to natural feature extraction because textural variations of the natural features cause spurious noise. Instead, mixed spatial frequency description of the textures are the most desirable texture descriptors and we have made use of a special type of filter, called Gabor filter, for texture representation [Daugman 1988]. An adaptive neural network to extract a Gabor representation of the image data is part of the first level image preprocessor routines.

At the second level, the neural network-based region feature extraction module has one or two stages of neural network processing depending on the kind of sensor fusion employed. In the case of data-level fusion, a collection of feed forward networks that use the Gabor filter responses from multiple sensors is arranged to overlook the entire image as a grid of cells to classify the image texture as forest, field, or water. In the case of decision level fusion, a second stage of processing is introduced to employ another set of feedforward networks to refining the decisions of networks from individual sensors to arrive at a collective decision on the image texture.

At the third level, the expert system is responsible for taking the output of the neural networks and refine the decisions made by the networks. In addition to analyzing the output of the neural networks, the expert system can interface with a geographic database. This interface is designed to operate in one of two modes: (1) using the database to improve feature extraction and recognition, and (2) refining the database using the output of the feature extraction system. The basic approach is to use the expert system to try to verify region boundaries as determined by the neural networks with region boundaries as determined by the geographic database. The database is intended only to serve as a guide to the system, since such databases are often outdated and the contents should be tempered by more contemporary evidence. The expert system could be easily extended to provide feedback to the neural networks which would enable the expert system to task the neural networks to do additional processing for feature extraction. Thus inconsistencies in high level feature extraction could be used to re-focus lower level feature extraction with appropriate modifications to the neural networks to incorporate the additional contextual information available.

5.2 Results of Testing

Only a tabular form of statistics on the performance of InFuse is provided here. Additional details and classification results along with the imagery are included in our final report [Raghavan et. al. N62269-90-C-0567].

	SAR without Fusion	Optical without Fusion	Data Level Fusion	Dec. Level Fusion	Expert Sys. Fusion
Forest	21.4 %	66.9 %	56.2 %	82.1 %	92.2 %
	65.2 %	76.4 %	86.7 %	94.3 %	92.5 %
Water	96.8 %	96.6 %	93.8 %	98.8 %	98.8 %
	96.8 %	98.3 %	99.6 %	99.6 %	98.8 %
Field	0.03 %	82.5 %	85.7 %	83.8 %	84.1 %
	0.06 %	91.7 %	94.6 %	84.2 %	92.7 %

Table 5.1: Results of Classification using InFuse

In Table 5.1, the first row in each category (forest, water, field) indicates the percentage of correct classification while considering rejects as misclassification. The second row in each category indicates the percentage of correct classification by excluding rejects. The results are promising with both types of approaches, although the decision level fusion performed slightly better than the data level fusion. The effect of sensor fusion is strikingly evident in these results. Efforts are underway to incorporate a textural boundary extraction method using a multi-resolution-based texture representation scheme.

6.0 Motion Computing and Multiple Object Tracking

Computing motion from an image sequence can be viewed from two different perspectives. In the first case, image motion is recovered at feature points of the image such as intensity changes. Known as correspondence-based methods, algorithms proposed from this perspective establish correspondence between points in the image sequence and recover motion from the relative displacement between corresponding points in the image frames. The major drawback of these algorithms is the need to solve the correspondence problem. In addition, for images with sparse intensity changes interpolation of velocities discovered at the feature points can be a formidable problem. Algorithms from the second perspective, also known as optical flow-based methods, compute image velocity at every point in the image. The vector field of image velocities at every point is termed optical flow or image flow. The major advantage of optical flow based algorithms is that they provide a dense velocity vector field without the need to interpolate. Such a dense velocity field can then be used to detect moving objects in the image. In this section, we briefly describe a motion computing and multiple target tracking algorithm designed for the Strategic Defense Initiative Organization (SDIO) in a Phase II SBIR effort from the Office of Naval Research (ONR).

In the literature, algorithms to compute optical flow begin with a useful constraint known as optical flow constraint [Horn and Schunck 1981]. Through the use of first principles of calculus it can be easily shown that the intensity derivatives at a point (x,y) in the image obey the equation:

$$I_x u + I_y v + I_t = 0$$

where $u = dx/dt$ and $v = dy/dt$ are the optical flow velocities at point (x, y) in the x and y direction

respectively and I_x, I_y and I_t are the intensity gradients with respect to x, y and t respectively.

The goal of image motion estimation algorithms is to exploit the optical flow constraint for a reliable and robust estimation of image motion. A number of algorithms in the literature use optical flow constraint in a regularization framework. Regularization-based methods originated with Horn and Schunck [1981]. The motivation for these methods comes from the observation that the optical flow constraint gives rise to only one equation for a pair of unknowns (u,v) at every point. It is obvious that we cannot recover u and v uniquely at every point since the system is under constrained. Additional constraints are required to estimate the velocity vectors uniquely. Since velocity vectors are expected to be continuous "almost everywhere", smoothness in estimated velocities can be added as an additional constraint.

The implication of adding the smoothness constraint is to involve an additional term in the objective function which is a measure of departure from smoothness,

$$e_s = \int (\Delta u^2 + \Delta v^2) dA$$

where the integral is over the image plane and Δ is the gradient operator. The deviation from the optical flow constraint can be written as

$$e_c = \int (\Delta I \bullet T + I_t)^2 dA$$

where $T = [u\ v]^T$. Horn and Schunck's algorithm minimizes $e_c + \lambda e_s$, where λ is a free parameter. The major disadvantage with this kind of method is that due to unconditional smoothing of the velocity field, these methods introduce incorrect velocity estimates where the velocity fields need to be discontinuous (e.g., boundaries of an object).

6.1 Discontinuities in Optical Flow

A serious difficulty with the smoothness constraint is that it tends to smooth the differences in image velocities at boundary points, making it difficult to separate them from other moving objects or the background [Raghavan et. al. 1992]. Therefore, it becomes difficult to estimate the object boundaries based on the smoothed image motion fields. While the assumption of motion field smoothness across object boundaries is not reasonable, the assumption of motion field smoothness within a body is both reasonable and useful. The assumption is reasonable because, on rigid bodies, the motion of one part of an object is highly constrained by the motion of nearby parts of the object. The assumption is useful because it can be used to reduce the effects of noise in computing the motion field.

Our approach to estimating a motion field with discontinuities is to simultaneously estimate the motion field and an additional set of variables which indicate the presence of discontinuities. This additional set of variables represents a line process [Blake and Zisserman 1987] and determines whether a discontinuity will occur between any two adjacent pixels. The basic idea in this approach

is that motion estimation can be considered to involve fitting of a smoothed velocity field to the field of estimated velocities at each point. In this fitting, a penalty is paid for a rapidly changing motion field. This penalty is directly related to the rate of change of the field. By allowing discontinuities in the motion field and not paying a penalty related to the rate of change in the motion at discontinuities, a better fit can be obtained. As a simple analogy, consider the problem of fitting a smooth curve to a step edge. To fit the step edge well, the smooth curve must change rapidly at the step and thus incur a large penalty. If discontinuities are allowed, then two horizontal lines will perfectly fit the step.

6.2 Computing Optical Flow with Discontinuities

It is clearly unreasonable to assume the presence of global continuity for optical flow. Instead, what is required is its "piece-wise continuous" nature. Marr [1982] introduced the term "continuous almost everywhere". While Marr's notion of continuity does not rigidly specify the location of discontinuities, piece-wise continuity constrains the possible locations of discontinuities. Blake and Zisserman [1987] suggested the weak constraint notion - a constraint which can be broken occasionally - to define weak continuity constraints for an appropriate class of continuous surfaces. The crux of weak continuity constraint is to prefer continuity while allowing occasional discontinuities if that makes for a simpler overall description.

The underlying objective is to construct a set of piecewise continuous 2-D functions u(x, y) and v(x, y) which satisfy the optical flow constraint well. This is achieved by modeling u(x, y) and v(x, y) as a weak elastic membrane - an elastic membrane under weak continuity constraint. Discontinuity occurs where the continuity constraint on [u v] is violated. The discontinuities can be viewed as a rupture or a break in the membrane.

The system of weak elastic membranes is specified by the its associated energy. The problem of finding optical flow then is reduced to minimizing that energy. The energy E for this system is

$$E = e_c + \lambda e_s + P$$

where,

$$P = aZ$$

where Z is some measure of penalty for introducing discontinuity and ais a weighting factor.

The quantity Z becomes the measure of the set of contours in the plane along which u(x, y) or v(x, y) are discontinuous. It might be defined in many ways and can include terms for 1) length of such a contour, 2) continuity of contour, 3) weak continuity constraint on contour smoothness and 4) cost for certain topological features. A detailed description for these can be found in [Blake and Zisserman 1987].

We used contour length as our measure of penalty. The reasons for that are two fold. First, this measure of discontinuity lends to computationally efficient algorithms. Any more complicated penalty term makes the problem, computationally almost intractable. Second, since Z is proportional to

contour length, contours of minimal length are obtained, which, due to hysteresis tend to be unbroken. The contour obtained can be subjected to subsequent processing for smoothing by applying the weak continuity constraint on contour curvature. The resulting objective function is nonlinear for which multiple minima may exist. We have used the Graduated Non-Convexity (GNC) algorithm of Blake and Zisserman [1987] to compute the global minima of this non-convex objective function. For reasons of limitations of space here we omit the details which appeared in the final report [Raghavan et. al N00014-89-C-0011].
The algorithm has been fully implemented and run on a number of test cases including moving random dot patterns as well as natural imagery. The results are quite encouraging and strongly indicative of the preservation of discontinuities in image velocities. A sample run of the algorithm on a moving random dot pattern is shown in Figure 6.1. For comparison, the result of running Horn and Schunck's algorithm is shown in Figure 6.2. A few major observations are as follows. (1) The discontinuities generated by the algorithm are potential cues which aid the image segmentation process. (2) The number of discontinuities allowed in the process is controllable on a global scale similar to multi-channel based discontinuity detection processes such as the Laplacian of the Gaussian. (3) A Hopfield type neural network can be used to minimize the objective function.

6.3 Neural Network-Based Segmentation and Tracking

Tracking of multiple objects in the image sequence can be done only if the objects in a given image frame can be segmented from each other. Motion cues in the form of image velocity vectors computed as before may not always be coherent in a region. Grouping these incoherent vectors to form connected regions is a challenging task. We have used a simple cooperative-competitive neural network to solve this difficulty. Such networks have been used before for computing stereo disparity [Marr and Poggio 1976]. The general underlying principle of this network is as follows. Neuron-like units are assigned for every spatial location with a specific preference to a motion direction in the image. For instance, a particular neuron may be sensitive to the image velocity lying between 45 and 60 degrees at a certain location in the image. The neurons are arranged, as shown in Figure 6.3, in a three dimensional structure such that the layers represent different motion directions in a contiguous fashion. A neuron is located at the intersection of a specific column corresponding to an image position and a layer corresponding to a specific direction of motion. The network uses similarity among the velocity vectors as a cue to "grow" a region, while using dissimilarity to terminate the growth. Consequently, neurons in the same layer excite each other locally in a Gaussian-weighted fashion, while neurons in the same columns inhibit each other. The segmentation is a result of cooperation and competition between the neurons of this three dimensional lattice. The neurons are threshold-type neurons and thus binary in nature. The network is updated in an iterative fashion and the process terminates when there is no change in the status of any unit of the network. We have found out that the segmentation results obtained from this network are quite robust although the results can be improved by employing another neural network with layers of speed sensitivity instead of directional sensitivity.

The next step is to associate a track with every potential target segmented in the image. This is s nontrivial problem. We make use of a number of cues including the size, shape, and image velocity represented as an affine transformation for this purpose. A numerical score is computed for every potential pairing of the targets appearing in the current frame and the previous tracks. The one with

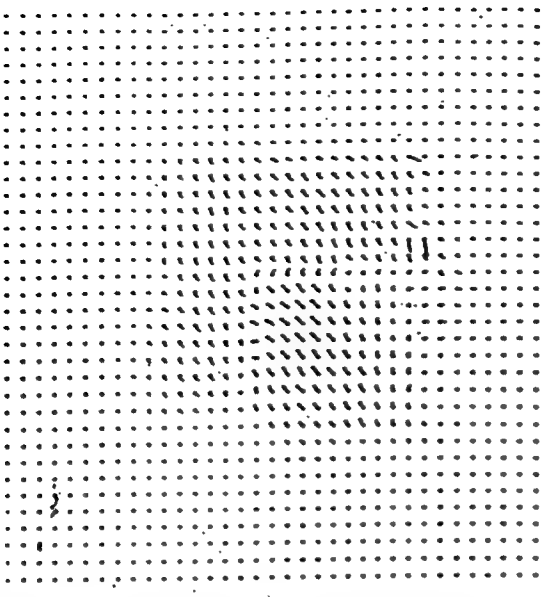


Figure 6.1: Results of Discontinuity-Preserved Optical Flow Computing on a Random Dot Moving Pattern

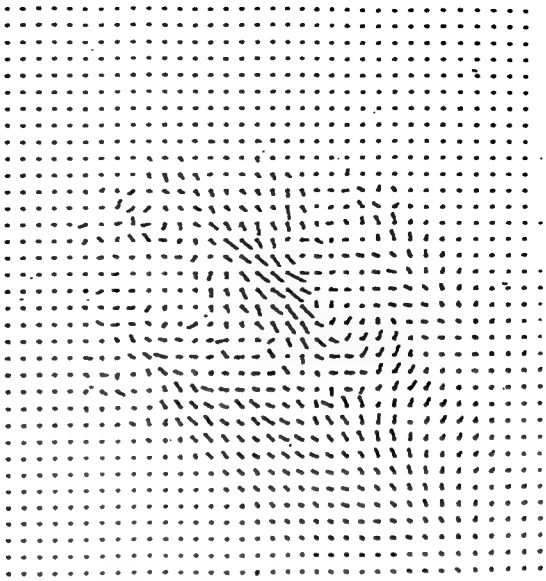


Figure 6.2: Results of Applying the Horn and Schunck's Algorithm on the same Random Dot Moving Pattern

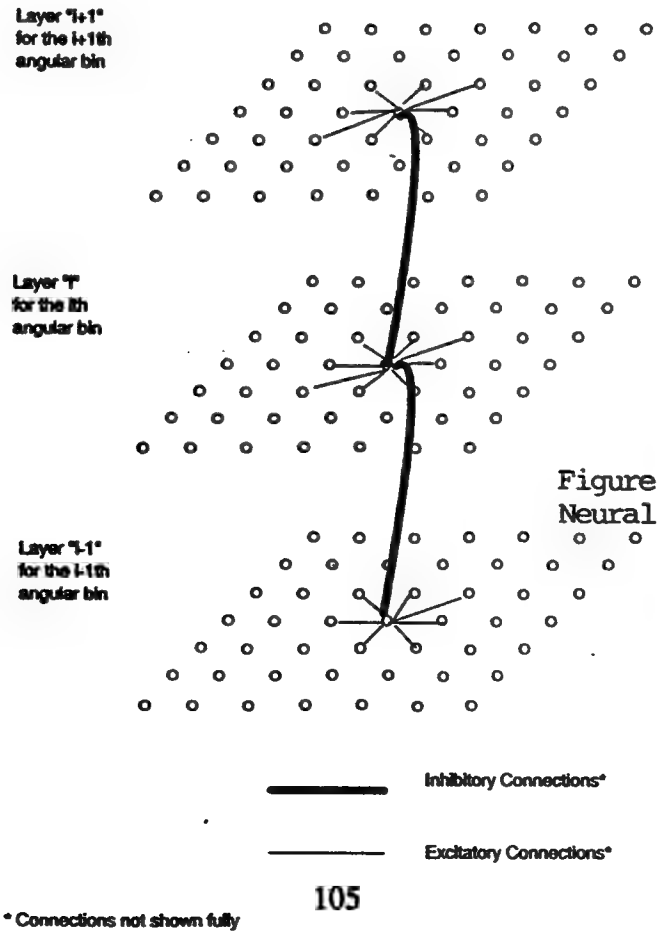


Figure 6.3 A Cooperative Neural Network for Segmentation

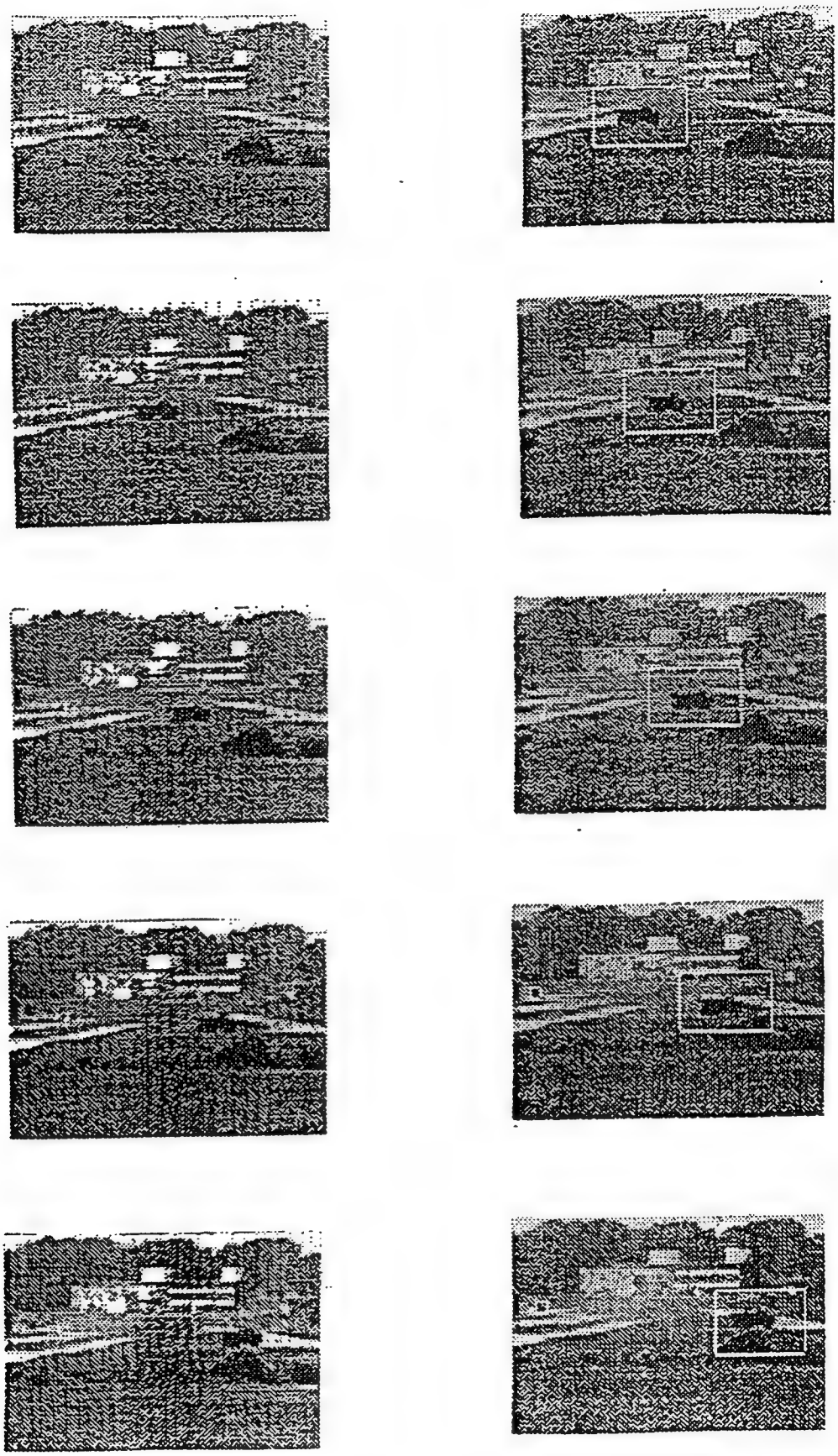


Figure 6.4: Results of Tracking on a Real Image Sequence

the highest score is selected and is used to update a Kalman filter assigned to every target. Some results of testing this scheme are shown in Figure 6.4. Clearly, we have left out a lot of details on the algorithms to compute optical flow, represent image motion, and segment and track the objects. The reader is referred to [Raghavan et. al. N00014-89-C-0011].

7.0 Conclusion

In this paper, we have provided an overview to the neural networks based Navy SBIR efforts completed at LNK. For additional details see the final reports by Lavine et. al. [N0024-90-C-3814], Bailey et. al. [N60921-89-C-0030], Raghavan et. al. [N62269-90-C-0567], and [Raghavan et. al. N00014-89-C-0011]. In every one of these effort we have found that neural networks offer the advantages of superior performance, simplicity and ease of use, flexibility, robustness, and parallelism. SBIR Efforts are currently underway to use this technology for other military applications such as location of relocatable targets and civilian applications such as fishery stock prediction.

8.0 References

[1] G. Bailey, N. Gupta, B. Lambird, D. Levine, J. Meyer, and S. Raghavan, "Classification of RCS Data Using Neural Networks", Final Report for SBIR Phase II Contract N60921-89-C-0030 Naval Surface Warfare Center, 1992.

[2] G. Bailey, S. Raghavan, N. Gupta, B. Lambird, and D. Lavine, "InFuse: An Integrated Expert Neural Network for Intelligent Sensor Fusion", Proc. of IEEE/ACM International Conference on Developing and Managing Expert System Programs, 1991.

[3] A. Blake and A. Zisserman, Visual Reconstruction, MIT Press, 1987.

[4] G.A. Carpenter and S. Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns", Applied Optics, vol.26, no.23, pp 4919-4930, 1987.

[5] J. Daugman , "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression", IEEE Trans. on ASSP, Vol. 36, No.7, July 1988.

[6] K. Fukushima,"Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition", Neural Networks, vol. 1, no. 2, 1988.

[7] R.P. Gorman and T.J. Sejnowski, "Analysis of Hidden Units in a Layered Network to Classify Sonar Targets", Neural Network s, vol. 1, pp 75-89, 1988.

[8] R. Hecht-Nielsen, "Applications of Counterpropagation Networks", Neural Neworks, vol. 1, no. 2, 1988.

[9] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley, Publishing Co., Reading, Mass., 1990.

[10] B.J. Hellstrom and Kanal, "Asymmetric Meanfield Neural Networks", To Appear in Neural Networks, 1992.

[11] B.K.P. Horn and B.G. Schunck, "Determining Optical Flow", Artificial Intelligence, vol. 17, pp 185-203, 1981.

[12] L. Kanal and S. Raghavan, "Hybrid Systems - A Key to Intelligent Pattern Recognition", Invited Paper, Proc. of IJCNN, Baltimore, 1992.

[13] K.J. Lang, A.H. Waibel, and G.E. Hinton, "A Time-delay Neural Network Architecture for Isolated Word Recognition", Neural Networks, vol. 3, pp. 23-43, 1990.

[14] D. Lavine, J. Meyer and B. Lambird, "Estimation Using Neural Networks", Final Report, SBIR Phase I, Naval Sea Systems Command, N0024-90-C-3814, 1990.

[15] D. Marr, Vision, W.H. Freeman Press, New York, 1982.

[16] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity", Science, vol. 194, pp 283-287, 1976.

[17] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, 1988.

[18] S. Raghavan, N. Gupta, B. Lambird, D. Lavine and L. Kanal, "A Layered Object Recognition System Using a Hierarchical Hybrid Neural Network Architecture", Proc. of SPIE Conf. on Advances in Robotic Vision, Cambridge, Mass., 1991.

[19] S. Raghavan, G. Bailey, D. Lavine, B. Lambird, and N. Gupta, "InFuse: Intelligent Sensor Fusion by Combining Neural Networks and Expert Systems", Tech. Report for SBIR Phase I Contract N62269-90-C-0567, Naval Air Development Center, 1991.

[20] S. Raghavan, N. Gupta, and L. Kanal, "Tracking and Recognition of Multiple Objects in Passive Imagery", Final Report for SBIR Phase II Contract N00014-89-C-0011, Office of Naval Research, 1992.

[21] S. Raghavan, N. Gupta, and L. Kanal, "On Disconituinity-Preserved Optical Flow", Proc. of 11th ICPR, Netherlands, 1992.

[22] J. Rajapakse and R. Acharya, "Multisensor Data Fusion within Hierarchical Neural Networks", IJCNN, Washington D.C., June 1990.

[23] D.E. Rumelhart et. al., "Learning Internal Representations by Error Propagation", in J.L. McClelland and D.E. Rumelhart, Parallel Distributed Processing, Vol. I and II, MIT press, 1986.

[24] G. Ruck, D. Barrick, W. Stuart, and C. Krichbaum, Radar Cross Section Handbook, Plenum Press, New York, 1970.

Neural Network Sensor Data Fusion Methods for Naval Air Traffic Control*

A.J. Maren, M.D. Lothers, R.M. Pap,
Accurate Automation Corporation,
Chattanooga, TN 37406;

R. Akita,
Naval Command, Control and Ocean Surveillance Center
RDT&E Division
Code 434
San Diego, CA 92152

Abstract: Sensor data fusion, involving data from multiple types of radars and other sensors is an aspect of improving naval air traffic control systems for aircraft carriers and amphibious ships [Pap et al., 1990]. The neural network technology can be applied to perform shipboard multisensor fusion of similar and dissimilar data for high confidence target identification.

The neural network concept is designed to fuse data from a variety of input sources such as Radars, IFF systems, Electronic Warfare Systems, Communication & Navigation Systems (e.g. JTIDS), and Command and Control Systems (e.g. NTDS). The system will fuse similar source data, (i.e. position)/velocity/acceleration), and dissimilar source data (i.e. frequency/prf/prt, etc.) to make a declaration of the identification of the source of the data.

A cooperative-competitive neural network is being used as a key component in a data association and fusion system for the tracking of interacting targets in a simulated noisy environment. Its architecture is well suited to recognizing relationships in images or data. Measures of the target kinematics were used in the neural network for data association. However since the cooperative-competitive neural network is a computationally complex algorithm, this neural network is reserved for situations when other techniques were unable to resolve the matching conflicts between target tracks and target detections. The output of this neural network is then combined with other sensor derived positions to create a refined estimate of each target track in the sensor fusion process.

---

* This work is sponsored by the Naval Command, Control and Ocean Surveillance Center, Code 434 (formerly Naval Ocean System Center) under a Small Business Innovation Research (SBIR) Phase II contract N66001-90-C-7021. The authors wish to express their appreciation to the Naval Air Systems Command, Codes PMA-213, AIR 51641, AIR 551-TA and Space and Naval Warfare Systems Command Code 2243 for their support of this work.

1.0 Introduction

The multi-sensor data fusion system using neural networks has been applied to these three major components; data association, data fusion, and target classification / identification. The sensor fusion challenges are defined in Waltz & Llinas [1990 and references contained therein]. They comprise two sets of task constraints. First, incoming target data from multiple sensors (possibly located in different positions and accessing data at different rates) must be correlated with existing master target tracks. In some cases, new master target tracks must be initiated. Tracks which have been inactive for a period of time and false alarms should be deleted from the active set of master target tracks.

The use of neural networks for sensor data association introduce an innovative approach to finding the best match between new targets and existing target tracks. This approach, using a multilayer cooperative-competitive neural network, provides a robust means for identifying the best one-to-one matches out of a set of possible many-to-many matches. It accomplishes this using a neural network which operates on the value of similarity metrics between potential matches out of two sets of objects. A combination of cooperative and competitive signal passing reinforces the final activation of neural network nodes corresponding to the best matches, while inhibiting the nodes corresponding to conflicting matches. This offers an advantage over traditional methods of identifying best matches, such as least squares.

Once the report-to-track association task is accomplished, sensor fusion can be done. The neural network approach focuses on fusing target kinematics, as other types of information (e.g. beacon identification, ESM signatures) are often uniquely obtained from their respective sensors and may be associated but not necessarily fused. Sensor data fusion is then one of combined association and fusion; new target reports are associated with the best possible master target track. Where possible similar information provided by the sensors and by target track positions are fused. A sequence of recent fused reports are associated with each track, and are updated periodically.

Display of fused target data is also a crucial component of this system. Our innovations in both report-to-track association and sensor fusion involve use of neural network algorithms.

2.0 Report-To-Track Association

Sensor data fusion leading to possible target identification, needs to work with each target as observed over time. The system uses report-to-track association, rather than static report correlation. Use of master target tracks allows prediction of where target locations will be, which will enhance report-to-track association.

2.1 Constraints Resulting from System Requirements Design and Analysis

Constraints governing the choice of the report-to-track algorithm selection and development:

- Work with multiple sensors: The methods selected is usable with several different types of sensors which may have different dimensionalities, different locations, different rates or types of target report access, etc. This suggests that the system should invoke several methods; i.e., report association based on beacon response or signature matching should be used where feasible as well as report-to-track matching using target and track kinematics.

- Computational throughput considerations: Advances in hardware and system software (including parallelizable systems) make possible more advanced algorithms, in a real-time environment. Nevertheless, the report-to-track association task has computational complexity on the order of $O(nxm_kxk)$, where n is the number of existing master target tracks and m_k is the number of new target reports from each of k sensors. Assume, for simplicity, batch processing of a sweep's worth of data as a group, where a typical sweep for a radar might be 4 seconds per cycle. Then all of the report-to-track associations need to be completed within about 4 seconds.

- Effective report-to-track matching under dense target conditions: The ideal approach to reconciling possible conflicts in report-to-track association under dense target conditions use as much information as possible in determining final associations. Optimally, a method can be found which allows the influence of additional target-descriptive information as well as target kinematics.

3.0 Cooperative-Competitive Neural Network

The cooperative-competitive neural network is well suited to several different applications. Its strength is that it is able to determine relationships between entities presented as input. Cooperative-competitive neural networks have been used with image understanding techniques since they are able to recognize important features in an image or in dynamic inputs, such as with automatic target recognition. Cooperative-competitive neural networks are also able to recognize one-to-one matches from many-to-many matches; a feature useful to target tracking [Maren et al., 1989 and 1990].

The cooperative-competitive neural network architecture is generally constructed with five logical layers of phases. Layer one stores initial inputs to the neural network. Layer two stores the current strength value of each node derived as a function of the input. The values in layer two are used to populate adjacent (competitive) nodes in the same neural network relation and excite corresponding (cooperative) neurons in other neural network relations. (Each relation may be a feature or type of data considered.) Layer three stores inhibitory or excitatory signals received from other neurons. Layer four applies these signals to the values in layer two. Finally, winning nodes are selected from the neural network indicating the most similar matches.

The cooperative-competitive neural network has been described extensively in Minsky and Maren [1990], Maren et al. [1990] and Minsky [1990]. A significant role in the neural network based sensor fusion system is performed by the cooperative-competitive neural network. It performs

final refinement of conflicting matches between target tracks and target detections. Once final refinements are made for each sensor-derived position, these values may be fused into target track kinematics.

4.0 Tracker

The neural network based sensor data fusion system has developed an adaptive approach to the classic alpha/beta tracker, which allows the alpha and beta coefficients to be updated in real time, using target kinematic information provided by the predicted target position and the sensor data. While the neural network which accomplishes this adaptive updating is a classic feedforward neural network, innovations include use of distance metrics and variances between these metrics as input to the neural network. Once trained, the neural network providing updates for alpha and beta operates very fast and is suitable for real time target tracking applications.

A modified version of the cooperative-competitive neural network, using a voting neural network, for target identification/classification is being investigated, as well as a backpropagating Perception to assess target classes. In each case, the input is provided by a variety of existing target classification/identification processes.

5.0 Algorithm For Data Association

Several processing steps are taken before the cooperative-competitive neural network is used. Polar coordinates of data coming from the bus are converted to x, y and z coordinates. if the target track has had three or more detections velocity and acceleration are available and a prediction is made for the location of the target track at the time of each new target detection. A gate is created around that predicted point. (Velocity may be used alone in the prediction, if necessary, but the gate around the target track is made larger to account for increased prediction error.) Any new target detection falling within that gate will be associated with this target track as a preliminary match. However, many conflicts may remain between the target tracks and target detections.

The cooperative-competitive neural network is applied when other techniques are unable to resolve conflicts between new targets detections and existing target tracks. Conventional techniques such as gates around the predicted position are first used. If IFF information is available from the sensor, this may be used to further refine these matches. The matrix created by the target tracks and target detections is partitioned only to include conflicts.

Several matrices are derived from the time period represented in Figure 1. A complete matrix showing all representations is shown in Figure 1(b). The partitioned matrices which will be passed to the cooperative-competitive neural network are displayed in 1(c). partitioning the pairing matrix illustrates how computational complexity is reduced.

Computational benefits are much greater as target tracks and detections increase. Assuming three sensors acquire the same targets in Figure 1, the number of operations the cooperative-

competitive neural network would use are 324 for the complete matrix. By partitioning the matrix only 117 operations for removing conflicts in target track 4 had no conflicts so the cooperative-competitive neural network is not required for this track.

6.0 Use of the Cooperative-Competitive Neural Network in Data Association

The cooperative-competitive neural network further refines the best pairing between existing target tracks and new target detections. One neural network relation is used for each dimension considered. A neural network is configured based upon the conflicting matches to be resolved.

Positional information, including the coordinates of the target detection and the predicted coordinates of the target track, is input into the neural network. The strength to each node is a function of the distance between the predicted position of the target and the actual position of the target detections. The distance between the predicted position of the target track and the target detection is inversely proportional, normalized between 0 and 1.

Inhibitions and/or excitations are used between neuron to determine a global winner in all the neural network relations. A strong neuron will inhibit competing neurons in the same neural network relation and excite corresponding neurons in another neural network relation. For example, corresponding neurons in y and z will have an excitatory effect on each other (i.e. the activation value of x will increase in proportion to the strength of y). The strongest neurons within the same x relation will have the strongest inhibitory effect on the other neurons. See Figure 2 for an illustration. The neuron with the strongest value among all neural network relations (x, y, and z) is selected as the winner and determined to be the closest target track/target detection pairing. The next strongest value will determine the next closest target tracks and target detection pair. This process will continue until no conflict exist between target tracks and target detections. If a target detection cannot be matched to a target, a new target track will be created.

For every target track/target detection matching problem, the cooperative-competitive neural network is reconfigured based upon the conflicts to be resolved. Neural network architecture is customized with the appropriate number of nodes based upon the target tracks and their potential matches in the pairing matrix. The strength of each neuron in the neural network is defined based upon the distance from its potential match.

For each sensor available, this data association with the cooperative-competitive neural network is performed, as needed. The output of the cooperative-competitive neural network represents the best position estimates for all target tracks from a single sensor. These position estimates from each of the J sensors must still be fused into a refined position estimate as shown in Figure 3.

7.0 Sensor Fusion

When data association has been completed, sensor fusion will be performed with a weighted sum of all different target positions. Based upon the confidence in each of the sensors, the values for

range, azimuth and altitude are combined to obtain to refined target position.

Confidence values for the weighted sum are a function of the past performance of the sensor reports or prediction estimates. Investigation of a tracker which will use a neural network (currently a modified backpropagation neural network with recurrent connections) to update the confidences in the predicted and sensor-derived positions is continuing.

Without adapting predicted and sensor-derived positions online, the alpha/beta tracker worked well in the simulated and testing environment.

This backpropagation neural network is actually an unsupervised neural network. Since the model position of the target is unknown in an actual target tracking scenario, error is derived from a measure of the distances between the sensor-derived positions and the predicted position. Training is an on-line process with the neural network updating confidences with the frequency of training specified by an operator. In many situations, training iteration(s) are not necessary on every scan.

8.0 Simulation Results

The results of this project were tested on a Silicon Graphics high performance workstation. Testing involved the use of simulated data. Data was created with the following assumptions: 1) Sensors are co-located. However, adjusting for offsets in sensor position will be straightforward. 2) Sensors are synchronized and have the same scan rate. Eventually the plan is to use multi-tasking in the Ada language to handle the asynchronous data from different sensors.

Our current simulator was written to create target tracks with three dimensional coordinates and calculate the time when the target made a change of course. The maximum rate at which targets could be detected was based upon the 1553 bus speed *(30 Hz). Time (bus cycle) at which the target is detected was determined by the speed of the target set in the simulator. The position of the target (expressed in polar coordinates) was set up graphically so that target tracks could be set up rapidly into interacting scenarios.

This simulated sensor data association and fusion system was tested with several scenarios. One scenario tested the crossing of two targets. Another scenario modeled two targets flying together and then splitting into different directions. Finally, a dog fight scenario was used.

Once the model path of each target was established, zero-mean Gaussian noise could be added to the target when needed. Adding standard deviations of 10, 15 and 25 percent zero mean Gaussian noise, did not negatively affect target tracking in any of these scenarios.

9.0 Conclusion

The use of neural networks for a multi-sensor fusion system can provide better performance than

comparable systems. The cooperative-competitive neural network has performed the tasks of sensor data association and sensor fusion in the system. Neural network architectures and training methods offer the flexibility and adaptability required to identify targets in rapidly changing environments. The neural network alpha/beta tracker offers some distinct advantages over conventional techniques. Together the sensor data association system with the neural network tracker with adaptable confidences and sensor fusion will be robust and efficient for future generations of naval air traffic control and radar systems.

10.0 References

Maren, A.J., Harston, C.T., & Pap, R.M. (1990). Handbook of Neural Computing Applications. San Diego: Academic Press.

Maren, A.J., Harston, C.T., & Pap, R.M. (1989). A hierarchical data structure representation for fusing multi-sensor information. Proceedings of the Second National Conference on Sensors and Sensor Fusion (Orlando, FL; March 27-31)

Minsky, V. & Maren, A.J. (1990). A multilayered cooperative-competitive network for segmented scene analysis. Journal of Neural Network Computing (Winter 1990, 14-33).

Minsky, V. (1990). A Multilayer Cooperative/Competitive Network for Grouping and Organizing Related Image Segments. Tullahoma, TN: University of Tennessee.

Pap, R.M., Harston, C.T., Maren, A.J., Parten, C. & Akita, R.J. Neural network implementation of multisensor fusion for target identification problems. Proceedings of the Third National Conference on Sensors and Sensor Fusion (Orlando, FL; April 16-20, 1990).

Waltz, E., and Llinas, J. (1990). Multisensor Data Fusion, Artech House.

Collision Avoidance Artificial Neural Network in Support of Mine Warfare

Michael G. Fisk
SEA Corp.

Michael Larkin
Prometheus, Inc.

Abstract: The Artificial Neural Network (ANN) which we are developing is essentially a collision avoidance algorithm to provide Tactical Decision Aid in the transition of a submarine through a mine field. The system will consist of an Artificial Neural Network which recognizes data patterns and makes recommendations, and a Display Processor which presents the data to the operator. The Display Processor will present the information to the user in a grid pattern. The grid values will be color coded based on the potential that a mine is in that particular space. The colors will range from red for a mine (i.e., a tracker has been assigned designating a mine), to blue designating that there is a minimum energy being received from active. White will be used to identify grids that the ship has already been through. To determine the potential of each grid containing something to avoid such as a mine, the Preprocessor will assign a value corresponding to the avoidance factor for that particular cube of ocean space. The Preprocessor will be an Artificial Intelligence based processor which will access, determine, and process avoidable data from sensors, users, and other ship's systems. This will be used as the data pattern to present to the Artificial Neural Network to recognize and recommend an appropriate maneuver. The ANN, itself, will essentially take the ocean grid pixels (voxels) and recognize contact patterns. The methodology used will be a system of weights to constrain satisfaction propagation and conjugate gradient descent method which is a variant of back-propagation which avoids local extreme.

By placing a preprocessor step on the front end allows the ANN to be applied to practically any problem requiring object avoidance or path determination. The following problems are examples two which this technology can be applied in solving collision avoidance:

- Autonomous, Unmanned Vehicles
- Robotics
- Merchant Ship Course Plotting (e.g., Valdez disaster)
- Airplane/Jet Course Plotting and
- Smart Highway Systems

******Final Paper Not Submitted. The Following Slides Were Used in Presentation.***

SBIR TECHNICAL DIRECTION

THE SPECIFIC APPLICATION OF ARTIFICIAL NEURAL NETWORKS (ANN) CONCERNING TACTICAL DECISION AIDS (TDA) FOR ANTI SUBMARINE WARFARE (ASW) IS IN THE ANALYSIS , INTERPRETATION, AND RECOMMENDATION OF TACTICAL DECISIONS CONCERNING AVOIDANCE OF SPECIFIC ELEMENTS. THIS METHODOLOGY WILL SUPPORT :

- *o COUNTER DETECTION PREVENTION,*
- *o ICE KEEL AVOIDANCE,*
- *o ENVIRONMENTS NON CONDUCIVE TO ASW PATROLLING, AND*
- *o MINE AVOIDANCE.*

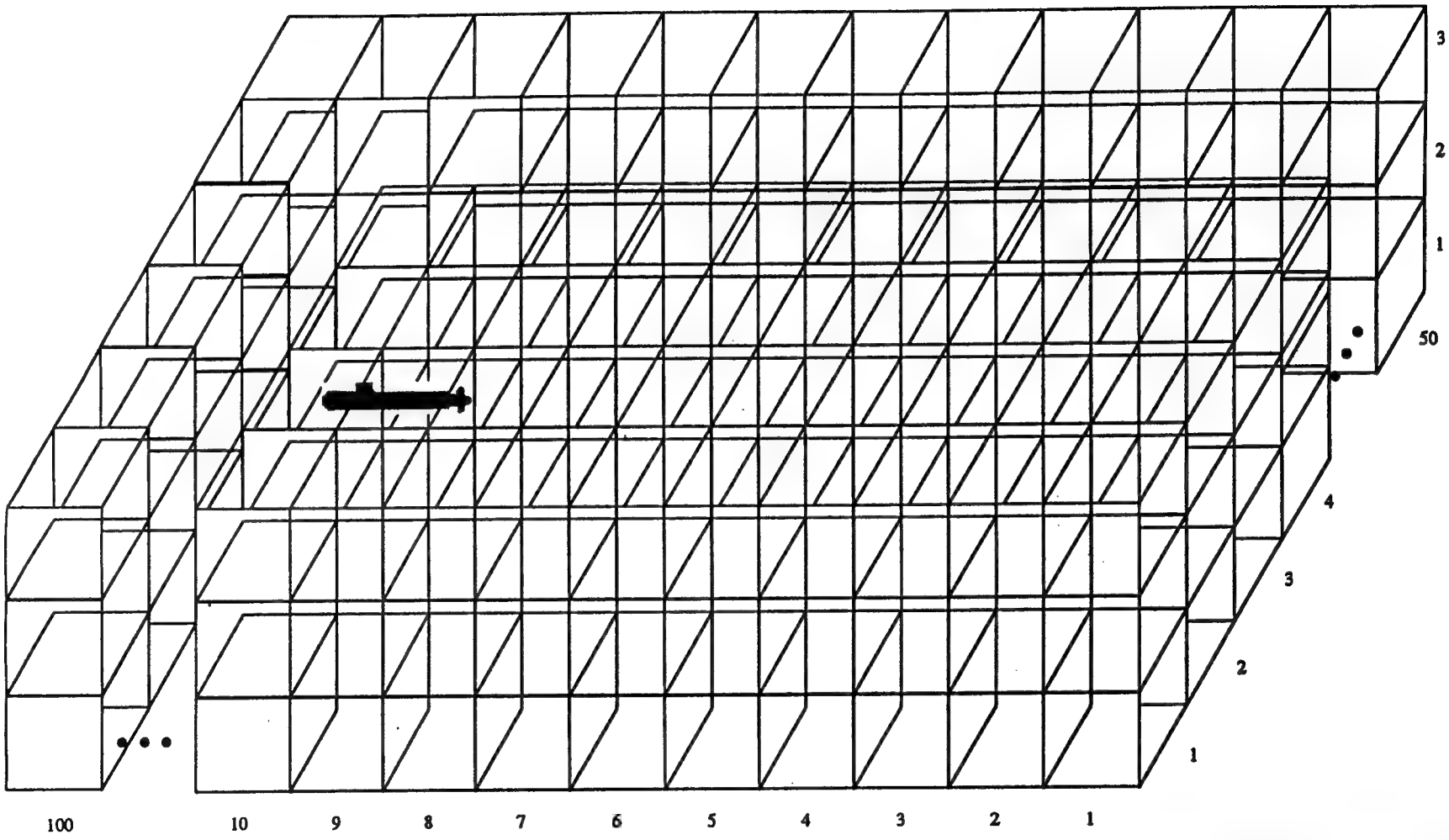
DUE TO THE RECENT CHANGES IN THREATS, AND DISCUSSIONS WITH NAVY LABORATORIES AND SQUADRONS WE ARE ADDRESSING THE MINE AVOIDANCE OF TACTICAL DESIGN AIDS AS THE PROTOTYPE SYSTEM TO INTRODUCE THE UTILITARIAN NATURE OF ARTIFICIAL NEURAL NETWORKS. THE NETWORK CONFIGURED FOR MINE AVOIDANCE TACTICAL DECISION AIDS WILL BE APPLICABLE TO THE OTHER ASW TDA REQUIREMENTS.

THE APPROACH AND TECHNICAL OBJECTIVES ARE STILL THE SAME, WE WILL:

- *o STEP 1 - ANALYZE INPUT/DATA,*
- *o STEP 2 - ANALYZE DECISION PROCESS,*
- *o STEP 3 - ANALYZE NEURAL NETWORK MODELS AND ALGORITHMS,*
- *o STEP 4 - ANALYZE NEURAL NETWORK HARDWARE, AND*
- *o STEP 5 - ARCHITECTURE DESCRIPTION.*

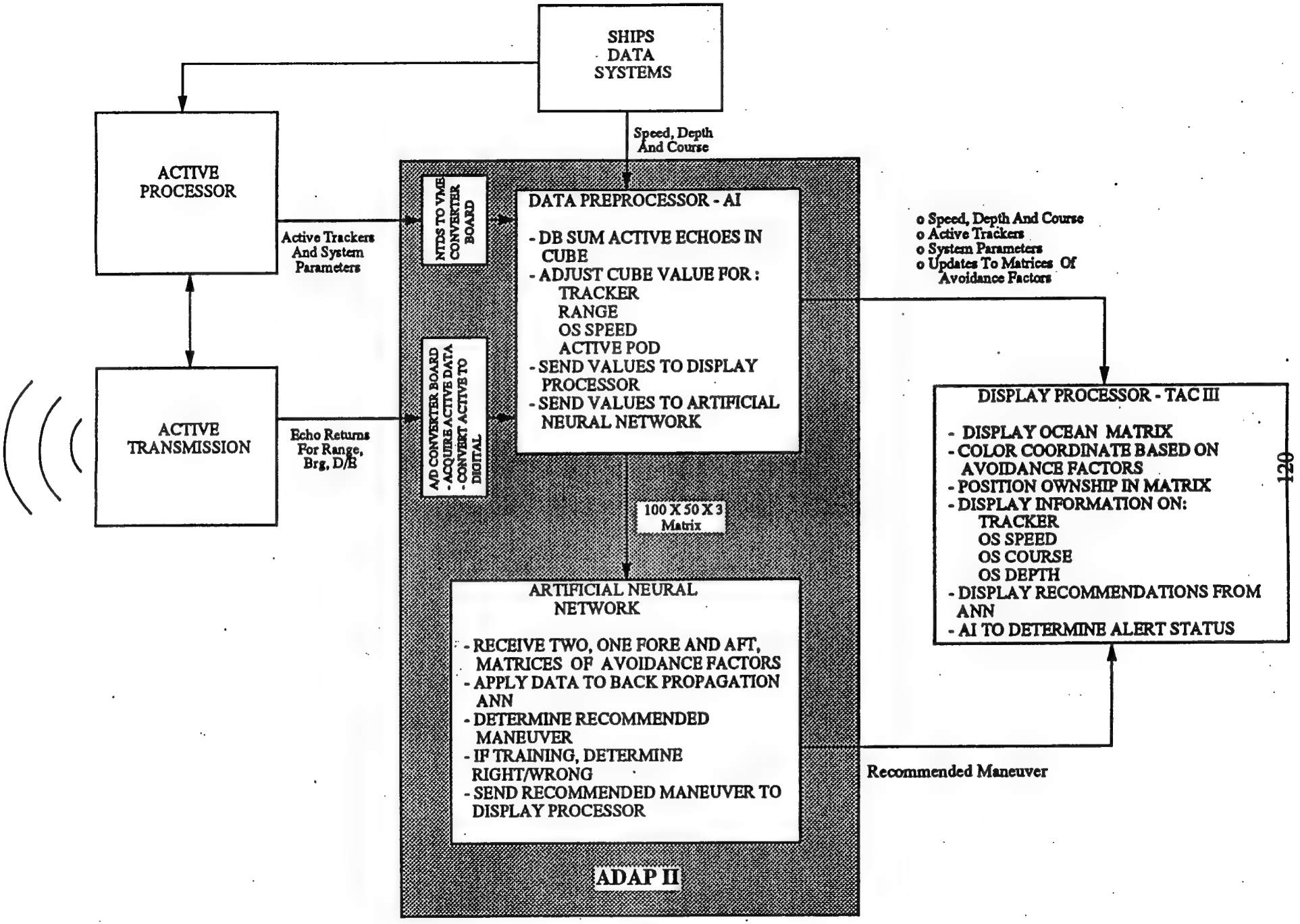
PROPOSED APPROACH

THE MINE AVOIDANCE TACTICAL DECISION AID TOOL WILL ASSUME THE SUBMARINE IS CENTERED WITHIN A MATRIXED OCEAN OF CUBES. THE THREE DIMENSIONAL MATRIX WILL CONSIST OF A HUNDRED BY FIFTY BY THREE GRID. EACH GRID WILL REPRESENT ONE THIRD (1/3) THE TURNING DIAMETER OF THE SUBMARINE WITH THE Z DIMENSION EQUATING TO THE SUBMARINES OPERATION DEPTH SEPARATED IN THIRDS.



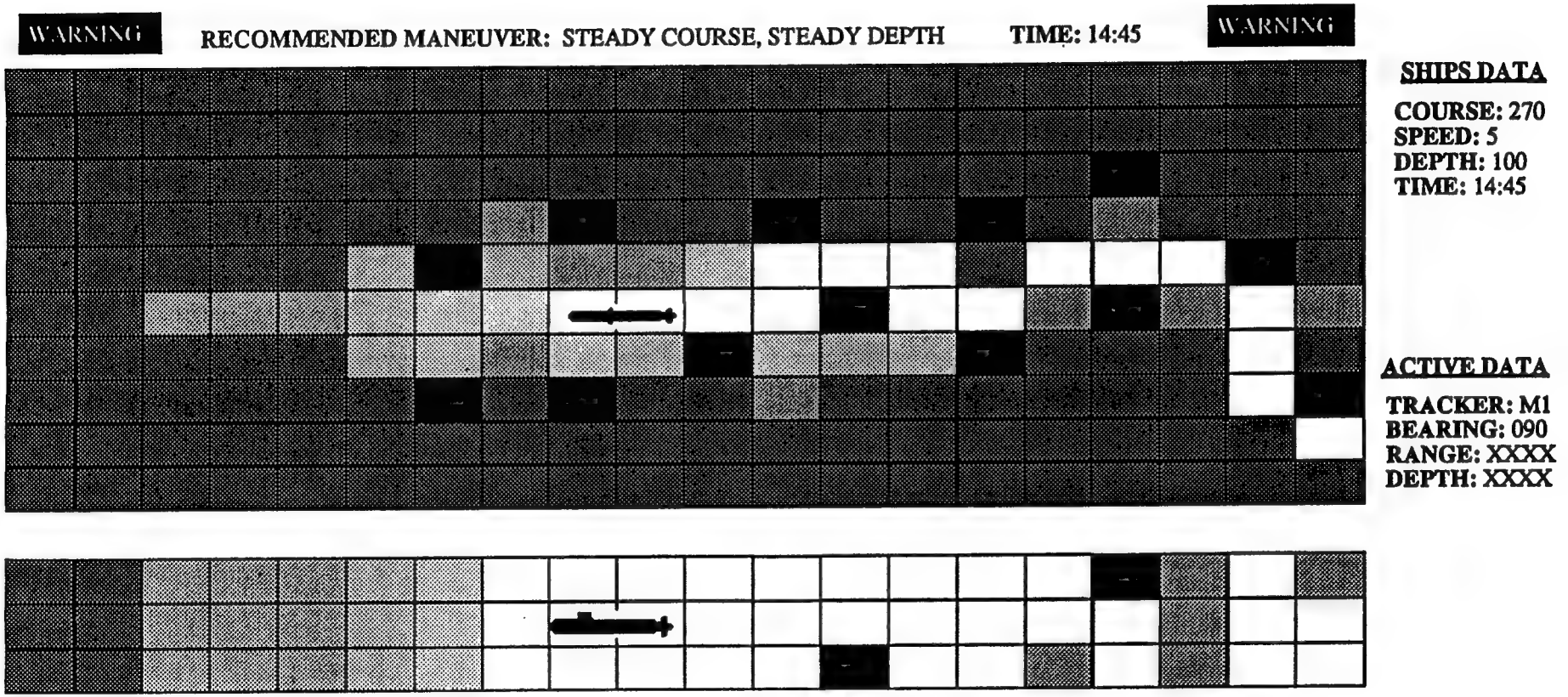
THE SYSTEM WILL CONSIST OF AN ARTIFICIAL INTELLIGENCE (AI) PREPROCESSOR THAT COLLECTS AND FORMATS REQUIRED DATA, AN ARTIFICIAL NEURAL NETWORK WHICH RECOGNIZES DATA PATTERNS AND MAKES RECOMMENDATIONS, AND A DISPLAY PROCESSOR WHICH PRESENTS THE DATA TO THE OPERATOR. THE CONFIGURATION WILL INVOLVE USING EXISTING EQUIPMENT SUCH AS THE ACOUSTIC DATA ACQUISITION PROCESSOR (ADAP II) AND THE TAC III COMPUTER.

PROPOSED FUNCTIONAL DESIGN



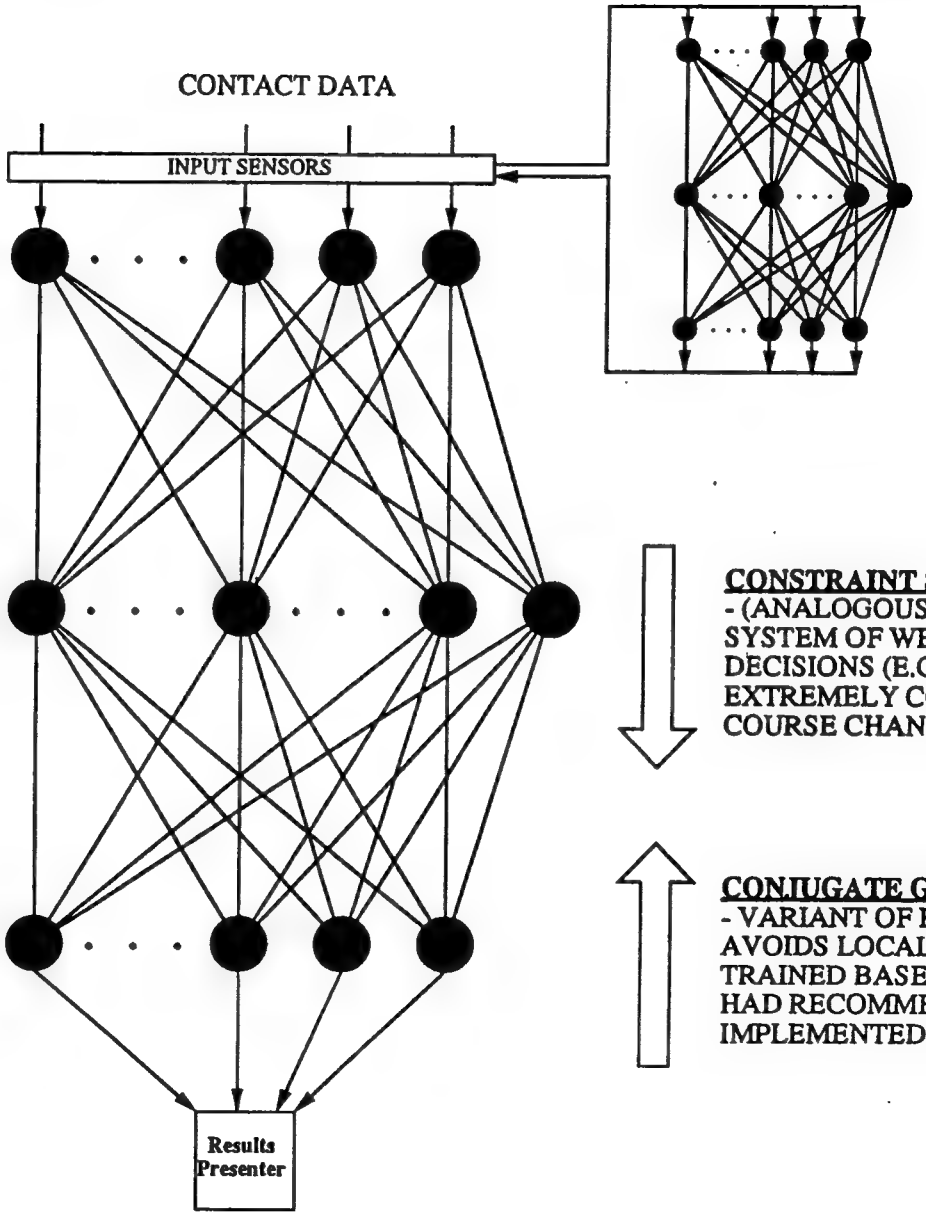
PROPOSED DISPLAY

THE MINE AVOIDANCE TACTICAL DECISION AID TOOL WILL PRESENT THE INFORMATION TO THE USER IN A GRID PATTERN. THE GRID VALUES WILL BE COLOR CODED BASED ON THE POTENTIAL THAT A MINE IS IN THAT PARTICULAR SPACE. THE COLORS WILL RANGE FROM RED FOR A MINE (I.E., A TRACKER HAS BEEN ASSIGNED DESIGNATING A MINE) TO BLUE DESIGNATING THAT THERE IS MINIMUM ENERGY BEING RECEIVED FROM ACTIVE. WHITE WILL BE USED TO IDENTIFY GRIDS THAT THE SHIP HAS ALREADY BEEN THROUGH. THE MATRIX IS SHOWN BELOW AS TWO DISPLAYS, WE WILL ATTEMPT TO GENERATE THE DISPLAY USING A SINGLE, THREE DIMENSIONAL MATRIX.



TO DETERMINE THE POTENTIAL OF EACH GRID CONTAINING SOMETHING TO AVOID SUCH AS A MINE, THE CUBES WILL BE ASSIGNED A VALUE CORRESPONDING TO THE AVOIDANCE FACTOR FOR THAT PARTICULAR CUBE OF OCEAN SPACE. THIS WILL BE USED AS THE DATA PATTERN TO PRESENT TO THE ARTIFICIAL NEURAL NETWORK TO RECOGNIZE AND RECOMMEND AN APPROPRIATE MANEUVER. THIS DISPLAY WILL BE CALCULATED USING AI RULES ON ACTIVE ECHOES, TRACKER ASSIGNMENT, PROBABILITY OF DETECTION VALUES, PREVIOUS LOCATIONS OF THE SHIP, OWN SHIP POSITION DATA, ETC.

PROPOSED ARTIFICIAL NEURAL NETWORK ARCHITECTURE



INPUT LAYER - NEURONS CORRESPOND TO PIXELS (VOXELS) IN OCEAN GRID

HIDDEN LAYER - REDUCE DIMENSIONALITY OF FEATURE SPACE; RECOGNIZE CONTACT PATTERNS STOCHASTIC CLASSIFIER - MODELS PATTERNS AS FINITE STATE IRREDUCIBLE MARKOV PROCESS

OUTPUT LAYER - DETERMINES/PROVIDES MANEUVERING RECOMMENDATIONS

OPTIONAL - CAPABILITY TO USE BACK PROPAGATION ANN TO CLASSIFY SONAR CONTACTS AS MINELIKE OR NON-MINELIKE

CONSTRAINT SATISFACTION PROPAGATION - (ANALOGOUS TO SCENE LABELING) USES A SYSTEM OF WEIGHTS TO CONSTRAIN DECISIONS (E.G., PRECLUDE CHOICE OF AN EXTREMELY CONVOLUTED SERIES OF COURSE CHANGES).

CONJUGATE GRADIENT DESCENT METHOD - VARIANT OF BACK PROPAGATION WHICH AVOIDS LOCAL EXTREMA. NETWORK IS TRAINED BASED ON SUCCESS OF MISSION HAD RECOMMENDED MANEUVERS BEEN IMPLEMENTED.

POTENTIAL COMMERCIAL APPLICATIONS OF COLLISION AVOIDANCE ARTIFICIAL NEURAL NETWORK

BY PLACING A PREPROCESSOR STEP ON THE FRONT END ALLOWS THE ANN TO BE APPLIED TO PRACTICALLY ANY PROBLEM REQUIRING OBJECT AVOIDANCE OR PATH DETERMINATION. THE FOLLOWING PROBLEMS ARE EXAMPLES WHICH THIS TECHNOLOGY CAN BE APPLIED TO IN SOLVING COLLISION AVOIDANCE:

- *AUTONOMOUS, UNMANNED VEHICLES,*
- *ROBOTICS,*
- *MERCHANT SHIP COURSE PLOTTING (E.G., VALDEZ DISASTER),*
- *AIRPLANE/JET COURSE PLOTTING, AND*
- *SMART HIGHWAY SYSTEMS.*

Autonomous

Control

Health Monitoring System for Aircraft

J. Gerardi and G. Hickman
Innovative Dynamics
Langmuir Labs Mail Stop 243
Ithaca, New York 14850

Abstract: Work is currently in progress to develop an advanced structural integrity monitoring system to increase safety of aging aircraft. This system is based on the concept of smart structures which integrates sensory systems into the structure, analogous to a central nervous system. Structural abnormalities are determined by continuously monitoring the vibration signature using a network of active sensor modules. Pattern recognition techniques are used to analyze the vibration signatures and identify structural damage in real-time. Conventional minimum distance algorithms as well as neural networks have provided high recognition rates in classification of corrosion damage and wing leading edge ice accretion.

1.0 Introduction

A Structural Integrity Monitoring System based on analysis of vibration signatures is being developed at Innovative Dynamics to detect structural abnormalities on aircraft. Current nondestructive evaluation techniques such as hand-held eddy-current or x-ray scans are so costly and time consuming that retiring some of the oldest jets may be more effective than maintaining them. An on-line inspection system such as SIMS holds the promise of solving these maintenance and diagnostics problems. The system described here consists of small surface-mount sensor module designs with integrated electronics that can be retrofit to existing aircraft. The objective is to integrate these modules into vulnerable or inaccessible areas of the airframe to reduce or eliminate the need for whole aircraft NDE scans or tear downs.

The principle underlying the operation of SIMS is the use of structural vibration signatures to determine mechanical properties. Damage to a structure often manifests itself as a change in the dynamic response of the structure, corresponding to changes in the physical properties. SIMS applies this concept for obtaining failure mode characterization of structural components. The system works by mechanically exciting the structure with broadband energy and monitoring changes in the structural response. Shape, amplitude and distortion of the vibration signals provide useful information concerning the location and severity of the damage.

Neural networks are attractive for vibration signature analysis. These techniques have been shown to be useful in solving complicated signal processing problems such as in NDE acoustic emission detection (Barga 1991). Neural networks require far

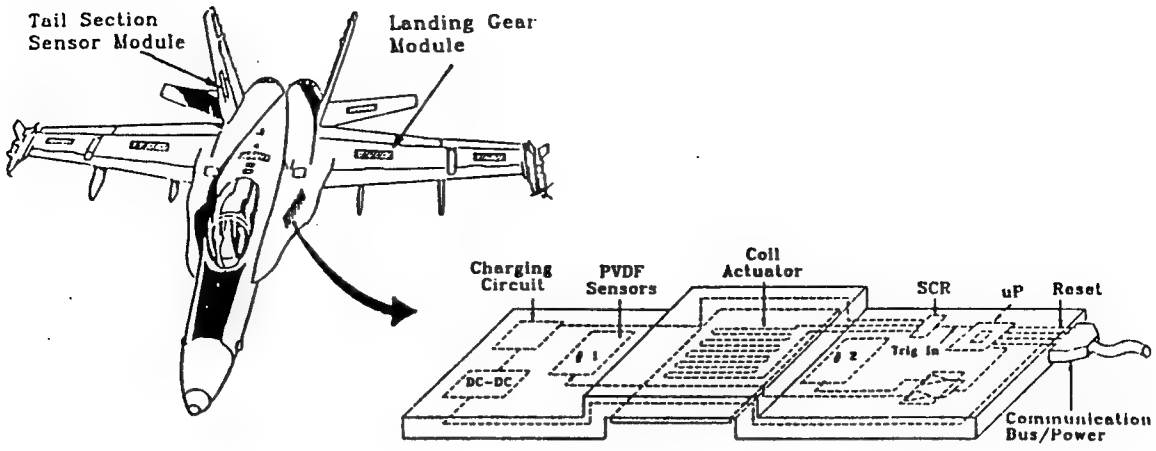


Fig.1 Structural integrity health monitoring system concept

assumptions about the structure of the input signal compared to other pattern classification techniques. In addition, the inherent parallelism of these networks allows very rapid parallel search and best-match computations required for monitoring complex aircraft structures.

Prototype system hardware and software have been developed and tested to determine the feasibility of a neural network based health monitoring system. Networks were trained to classify signatures from representative aircraft structures with simulated rivet line corrosion and with ice accretion. The neural network performance data are compared to that of the traditional nearest neighbor classifier used in earlier studies (Hickman 1991). This classifier serves as a good performance benchmark since it can be used to obtain upper and lower bounds on the Bayes probability of correct classification as the number of sample signals increases (Cover 1967).

2.0 System Architecture

Key components of SIMS are smart sensor modules daisy chained to a host central processor via a serial data communications link as depicted in Figure 1. The host processor interrogates individual structural components which contain the attached or embedded sensor modules that then relay digitized vibration signatures back to the host computer. The modules contain several piezoelectric vibration sensors, a pair of 12 bit microcontroller chips, an eddy-current actuator, and associated power and signal conditioning electronics. A network of these modules serves as a nervous system in detecting and recording the health of the structure. The eddy-current provides the impulse excitation source to the structure. When a pulse of current is sent through the eddy coil, currents are induced in the metal skin of the structure, creating a repelling force or impulse. A dual coil version can be used for composite structures. Excitation energy is on the order of 1 to 10 Joules, depending on the size of the structure. The piezoelectric sensors detect the dynamic structural response. Once the signals are processed by the host computer, diagnostic information is stored and the address code of the next module is selected on the bus. After all the modules have been interrogated, the data is displayed and/or removed from the

aircraft processor with a removable storage disk for further analysis and routine maintenance logging procedures.

3.0 Neural Network Algorithm

The neural network used was a multilayered perceptron trained using backpropagation learning (Rumelhart 1986). The network is composed of three layers of processing elements that perform a nonlinear trans-formation on their summed inputs and produce continuous-valued outputs between -1 and 1. A schematic diagram of the network is shown in Figure 2. A number of signal representations could be used as input to the neural network. The simplest would be to use the digitized waveform directly. For this study, however, a feature extraction procedure was used before processing with the neural network for easy comparison with previous results obtained using the nearest neighbor classifier. This procedure consists of generating an appropriate set of features for discrimination between classes. Both time and frequency domain parameters including damping ratio, peak amplitude, and spectral energy in different frequency bands have been found useful for vibration signal analysis (Hickman 1991).

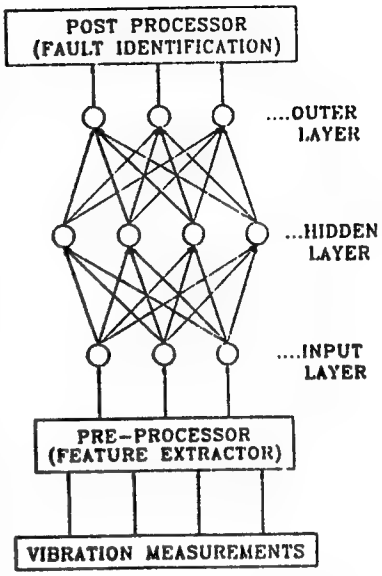


Fig. 2. Neural network architecture

4.0 Classification Experiments and Results

Experiments were designed to determine the ability of a neural network based diagnostic system to identify corrosion damage and ice accretion. Training the network consisted of repeated presentations of input-output pairs representing the damage case to be learned. The trained network was presented with a set of test returns excluded from the training set to determine its ability to generalize.

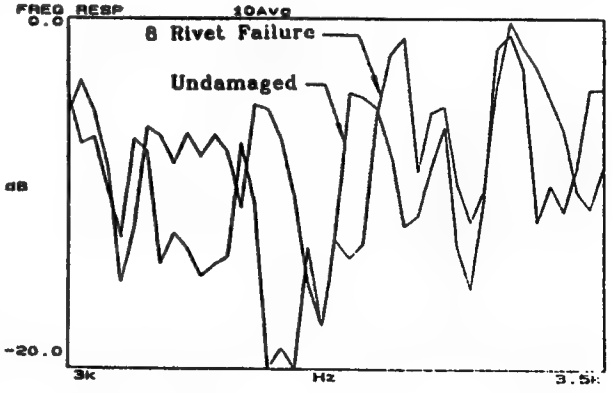


Fig.3. Typical sensor frequency response showing baseline (undamaged) and an 8 rivet failure case.

Laboratory experiments were performed on a 24" square 0.080" thick aluminum plate. Aircraft screws were used to clamp the edges of the test panel to a jig which provided rigid support to the test panel. A sensor module was attached to the bottom center of the test panel. The system was trained to recognize simulated corrosion by loosening a series of 4 and 8 consecutive

screws at different locations around the plate. To illustrate the variation in vibration signatures, Figure 3 displays the frequency response of the plate without any damage and with 8 loosened screws. The neural network results were outstanding, 100% of the responses were correct in indicating the severity of the damage. The nearest neighbor classifier did not fair as well, 84% of the responses were correct. Of note is the sensitivity of the nearest neighbor to proper selection of the input features. If two nonrelavent features were also used, the nearest neighbor performance dropped to 62% while the network performance was unaffected. Work is currently in progress to reliably identify the location of the damage as well as severity.

Sensor modules were also installed on the inside surface of a prototype leading edge wing section for in-flight testing (Hickman 1990). This prototype wing section is a 50" long wing cuff or glove designed to slide onto the DHC-6 Twin Otter main wing. The system was initially trained in the NASA Lewis icing research tunnel. Ice was allowed to build up continuously on the wing cuff and the system was trained in increments of 0.05 inch of ice up to a maximum thickness of 0.5 inch. Figure 4 displays one of the features that was used in the pattern classification. This figure shows the energy present in the frequency band 1150-1800 Hz as the ice thickness increases. Corrosion was also simulated by loosening aircraft screws as was done in the laboratory tests. Once trained, the system was tested in-flight. Comparable performance was observed using the network and nearest neighbor classifier, 94% of the responses gave the correct ice thickness. Corrosion was also reliably identified, with results similar to those obtained in the flat plate laboratory experiments. These flight tests demonstrated the capability of the system to duplicate results in the high noise environment of turboprop aircraft.

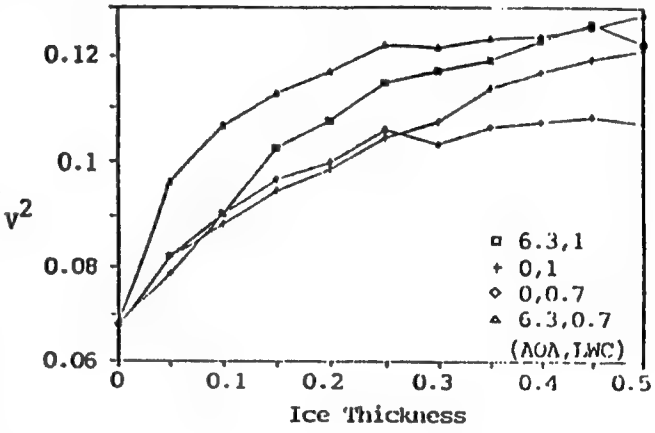


Fig.4. Sensor response on wing cuff during NASA Lewis icing tunnel test (partial power in 1150-1800 Hz for different angle-of-attack and liquid water content)

5.0 Conclusions

Smart structures comprised of attached sensor/actuator modules were found to be highly effective in gathering structural vibration data with good S/N and acceptable size, weight, and power requirements for aircraft applications. Initial laboratory and flight tests showed the system to perform well in identifying structural abnormalities using pattern recognition techniques. Work is in progress to develop a damage assessment methodology based on analytical and experimental modal analysis to reduce the training procedure. A concurrent effort is devoted to develop neural network topology that will have the ability to recognize and immediately incorporate new input data that fall into classes for which the network has

flight tests showed the system to perform well in identifying structural abnormalities using pattern recognition techniques. Work is in progress to develop a damage assessment methodology based on analytical and experimental modal analysis to reduce the training procedure. A concurrent effort is devoted to develop neural network topology that will have the ability to recognize and immediately incorporate new input data that fall into classes for which the network has not been trained.

6.0 References

Barga R S and Meador J L 1991 *SPIE* 1469 pp 602 - 611.

Cover T M and Hard P E 1967 *IEEE Trans on Information Theory* IT-13 pp 21-27.

Hickman G, Gerardi J and Feng Y 1991 *Journal of Intelligent Material Systems and Structures* 2(3) pp 411-430.

Hickman G, Gerardi J, Feng Y and Khatkhate A 1990 *ID Report NAS3-25200.*

Rumelhart D E and McClelland J L 1986 *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (MIT Press).

A Decentralized Adaptive Joint Neurocontroller

Cox, C.J.(1), Lothers, M.(1), Pap, R.M.(1,2), & Thomas, C.R.(1,3)

1. **Accurate Automation Corporation; 1548-B Riverside Drive; Chattanooga, TN 37406**

2. **The University of Tennessee, Center for Neural Engineering Applications; Knoxville, TN 37996**

3. **Covenant College; Lookout Mountain, GA 30750**

Abstract: This paper describes a decentralized adaptive joint neurocontroller for an n-joint robot arm. The joint neurocontroller consists of a PVA controller and a PD controller whose terms are adapted by recurrent functional link neural networks. It uses positional information only. Simulation studies of a two joint robot arm have shown that this controller is stable and robust. Our joint neurocontroller performed as well or better than one current adaptive controller technique and one current nonadaptive controller technique.

1.0 Introduction

Our joint neurocontroller is part of a larger general neural system that we are designing for the control of robotic manipulators [Pap, et al, 1991; Parten, et al, 1990a; Parten, et al, 1990b; Parten, et al, 1991]. This system [Pap, et al, 1991; Parten, et al, 1990a; Parten, et al, 1990b; Parten, et al, 1991] is being developed under NASA and ONR funding. Our objective under the NASA contract is to improve the operation of the Remote Manipulator System (RMS) on board the Space Shuttle. Under the ONR contract, we intend to control underwater manipulators. This research will result in actual neurocontroller hardware controllers. We envision two possible scenarios in which artificial neural network's could be used to aid these operations:

1. Artificial neural networks could be used to smooth and impose bounds on the possible movements initiated by the operator.
2. Artificial neural networks could be used for semiautonomous control, i.e. the operator specifies actions and the controller does the rest.

We have decided on a functional design which will be capable of either of these levels of operation. At the highest level is path planning. At the mid level is inverse kinematics. At the lowest functional level is joint control. We intend to develop neural network controllers (hence "neurocontroller" [Werbos, 1990]) for each of these functions. A neurocontroller for joint control have already been developed under NASA funding, utilizing the Accurate Automation Neural Network Toolbox. These joint controllers are the primary focus of this paper.

Our joint controller is based on concepts originally developed by Seraji [1989]. Like Seraji, we

adapt the coefficients of conventional linear controllers. Unlike Seraji, who uses conventional integration methods, we have chosen to explore the use of neural networks to adapt the coefficients [Pap, et al, 1991; Parten, et al, 1990a; Parten, et al, 1990b; Parten, et al, 1991]. Our neurocontroller retains all of the advantages of Seraji's approach:

- A complex model of the remote manipulator dynamics system is not required;
- Each joint can be controlled by a separate, independent controller;
- Each joint controller depends only on the current joint angle and the desired or reference joint angle;
- The decentralized scheme lends itself to a parallel implementation;
- Global asymptotic stability of the control system is assured.

The neural networks of our controller allow for greater parallelism than Seraji's approach. Serial simulations on a Silicon Graphics workstation have indicated that, even without this advantage, our controller's performance and speed are equal to or better than Seraji's method with Runge-Kutta integration.

Our simulation studies were run on our graphical robotic test-bed software. This general graphical test-bed is useful for analyzing various robot manipulator control designs. This software runs on the Silicon Graphics line of workstations, including the 4D/20, 4D/35 and 4D/340.

2.0 Controller Design

Our controller consists of three main parts. The first is a feedforward proportional-velocity acceleration controller (PVA). The second is a feedback position-derivative (PD) controller. The third part consists of six recurrent functional link [Pao, 1989] neural networks. Five of these compute the coefficients of the PVA and PD controllers. One computes an auxiliary signal. This structure is shown in Figure 1.

Our neural network implementations are second order functional links. Each of five of the networks calculate a single term of the PVA or PD controllers. The sixth network computes an auxiliary term which is added to the output of the PVA-PD controller combination.

All six of the networks are identical in structure. They differ only in their weight sets. This continuity in the design will allow for an easy transition to hardware in the future. As input, each network receives a vector consisting of three delayed reference joint angles and four delayed actual joint angles. Each network computes the thirty-five non-redundant terms of the outer product of the input vector and the input vector with a trailing component of one added. The dot product of the resulting vector and the network's weight set is computed. This scaler value is added to the last output of the output node, giving the output value of the network at the current time step. This output is interpreted as a torque value. The network structure is shown in Figure 2.

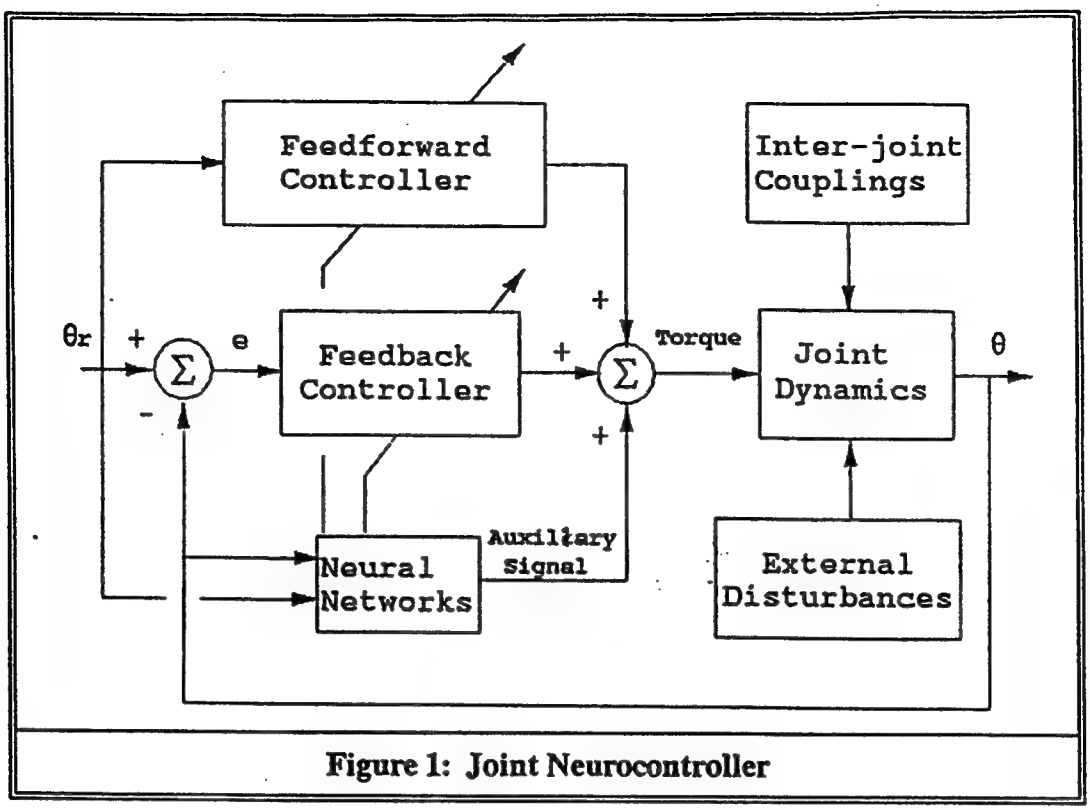


Figure 1: Joint Neurocontroller

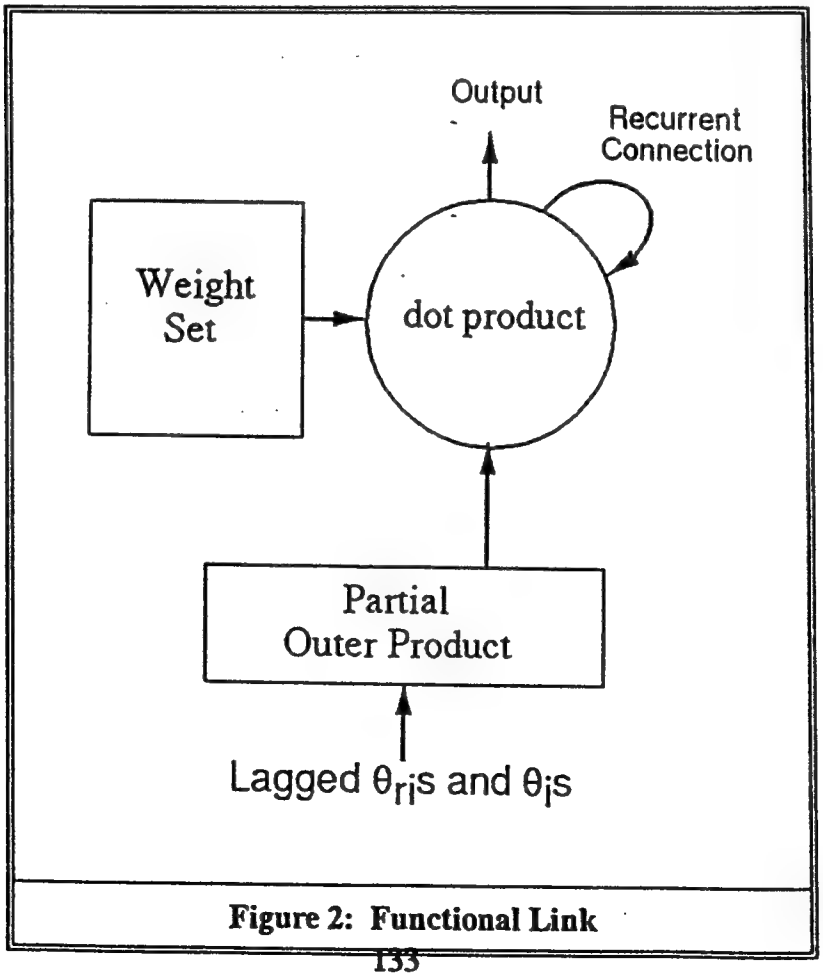


Figure 2: Functional Link

The networks can be trained based upon the observed output of a similar controller. It is also possible to calculate a weight set for each so that an appropriate integration is performed. We have opted to calculate the weight sets for our test simulations. In each of the networks, there are several weights which are zero. These could be left out, resulting in fewer computations and faster serial execution. We have chosen to retain these weights in order to keep the structures of the networks the same. This will allow us to easily transition these networks to identical parallel hardware in the future.

3.0 Experiments/Comparison

We conducted our tests on our graphical robotic test-bed on a Silicon Graphics workstation. All computations were performed serially. Motion was limited to a single plane. The simulation parameters were obtained from Seraji [19891, Fu, et al [1989], and Slotine [1986, 1991]. The modeled parameters include viscous frictions, Coulomb frictions, gravity loading, inertia of the links, link lengths, coupling effects, and end-effector payload. Table 1 gives the values. The angle of the first link was measured with respect to the x axis. The angle of the second was measured with respect to the first joint.

PARAMETERS SET FOR PUMA 560 ROBOT ARM			
m1 = 15.91 kg			
m2 = 11.36 kg			(link mass)
l1 = l2 =		0.432m	(link lengths)
V1 = V3 =		1.0 Nt m/rad s	(viscous friction)
V2 = V4 =		0.5 Nt m	(Coulomb friction)
Model Parameters			
a1 =	3.82		
a2 =	2.12		
a3 =	0.71		
a4 =	81.82		
a5 =	24.06		

Table 1

We compared the performance of five controller configurations. These were non-adaptive decentralized control [Tarokh, 1989], decentralized adaptive control [Seraji, 1989] using Runge-Kutta integration, and decentralized adaptive control using neural networks at three different weight levels. These levels were the original, 5% of the original, and 1000% of the original. Each controller was simulated with all the terms of its most general form. We tested the controllers' abilities to follow cycloidal trajectories from {90,0} to {0,90} in two seconds. In order to test the robustness of the controllers, end-effector payloads of 18kg and 100 kg were dropped on the arm at 0.5 seconds and released at 1.5 seconds.

NO PAYLOAD					
	Torque 1	Torque 2	Error 1	Error 2	Time
D.A.C	132.95	41.56	0.2235	0.1203	7
N.N.	143.44	48.89	0.5042	0.2177	4
N.N.L	114.94	40.41	4.7269	1.6272	4
N.N.H	228.38	73.00	0.1089	0.0974	3
D.C.	81.43	28.50	1.2834	0.5042	3

PAYLOAD - 18 kg					
	Torque 1	Torque 2	Error 1	Error 2	Time
D.A.C	432.03	217.04	0.4584	0.3380	8
N.N.	410.49	209.61	1.4209	0.6761	3
N.N.L	295.53	68.28	10.7258	6.2853	3
N.N.H.	538.88	256.93	0.2120	0.1432	4
D.C.	217.83	122.90	4.3602	2.8361	2

PAYLOAD - 100 kg					
	Torque 1	Torque 2	Error 1	Error 2	Time
D.A.C.	NaN*	NaN	NaN	NaN	43
N.N.	1401.83	817.19	4.4805	2.5898	3
N.N.L.	1022.66	723.80	26.9519	19.1311	3
N.N.H.	1916.27	1032.66	0.6303	0.3896	3
D.C.	1042.24	688.69	20.0191	13.8427	2

* NaN used to represent infinity

Torque 1: Maximum torque produced at joint 1 in Newton meters
Torque 2: Maximum torque produced at joint 2 in Newton meters
Error 1: Maximum error of link 1 in degrees
Error 2: Maximum error of line 2 in degrees
Time: Duration of simulation in seconds

D.A.C. Decentralized adaptive control using fourth order Runge-Kutta
N.N. Decentralized adaptive control using neural networks
N.N.L. Neural networks with weights distorted to only 5% of their original value
N.N.H. Neural networks with weights distorted by 1000% of their original value
D.C. Non-adaptive decentralized control

Table 2

First, we compared the controllers by moving the arm from {-90,0} to {0,90} with no payload. The payload was never varied during this first motion. Both the standard and neural decentralized adaptive controllers were very accurate. The neurocontrollers quickly stabilized the arm so that the error went asymptotically to zero. The Runga-Kutta integration of the standard version resulted in a continuous wobble about zero positional and velocity errors. The nonadaptive controllers were much less accurate than the adaptive controllers.

The robustness of each controller was tested by dropping and adding weights to the end effector during a motion. A 100kg weight dropped onto the end-effector at 0.5 seconds did not significantly effect the positional accuracy of the neurocontrollers. When it was dropped off at 1.5 seconds, the error stabilized. The integration controllers failed. Their output went to infinity when the weight was released. The nonadaptive controllers did not fail this test, but positional and velocity errors were large.

The results of these and further tests are summarized in Table 2.

4.0 Summary

We have developed a decentralized adaptive joint neurocontroller for an n-joint robot arm as part of a general semiautonomous telerobotic neurocontrol system. This joint neurocontroller utilizes functional link neural networks. It has shown itself to be accurate and robust. It is equal to or better than two current techniques of decentralized control, that of Seraji [1989] and that of Tarokb [1989].

5.0 References

Pao, Yoh-Han (1989). *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley.

Pap, R.M., Parten, C.R., Rich, M.L., Lothers, M., and Thomas C.R. (1991) Underwater Robotic Operations Using A Decentralized Adaptive Neurocontroller. Presented at *The IEEE Conference on Ocean Engineering.* Washington, D.C.

Parten, C.R., Thomas, C.R., Pap, R.M. and Lothers, M. (1991). A decentralized adaptive neurocontroller for robotic operation. Presented at *The Second Workshop on Neural Networks: Academic/Industrial/NASA/Defense (WNN-AIND 91).* Auburn, AL.

Parten, C.R., Pap, R.M, & Thomas, C. (1990a). Neurocontrol applied to telerobotics for the space shuttle. *Proceedings of the International Neural Network Conference* (Vol 1, pp. 229-236).

Parten, C.R, Pap, R.M, Harston, C.T., Maren, A.J., Rich, M.L., & Thomas, C. (1990b). Application of neural networks for telerobotics as applied to the space shuttle. *Proceedings of the First Workshop of Neural Networks: Academic/Industrial/NASA/Defense* (pp. 285-302).

Seraji, Homayoun (1989). Decentralized Adaptive Control of Manipulators: Theory, Simulation and Experimentation. *IEEE Transactions on Robotics and Automation* (Vol. 5, No. 2).

Slotine, Jean-Jacques E. & Li, Weiping (1991). *Applied Nonlinear Control.* Englewood Cliffs, NJ: Prentice Hall.

Slotine, Jean-Jacques E. & Asda, H. (1986). *Robot Analysis and Control.* New York: John Wiley and Sons.

Tarokh, M. (1989). Simulation Analysis of Dynamics and Decentralized Control of Robot Manipulators. *Simulation.* October.

Werbos, Paul J. (1990) Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE* (Vol.78, No.10, pp. 1550-1560).

Robotic Non-Destructive Inspection of Aircraft for the Navy

Dan Greenwood, Netrologic
Spenser Menlove, Netrologic
Dr. Ernest Franke, Southwest Research Institute
Dr. Mahmoud Tarokh, San Diego State University

Abstract: A feasibility analysis of the application of robotics to the Non-destructive Inspection (NDI) of aircraft found that it is feasible to transform the Navy's robotic deriveter to a neural network based NDI robot. A feedforward multilayer neural network was very reliable at detecting cracks around rivets and an efficient new manipulator path planning method using neural networks was found to be useful for the robotic aircraft NDI.

1.0 Introduction

Non-Destructive Inspection (NDI) of aircraft is known to be a time consuming, boring, and an error prone task for human inspectors despite many advances in NDI sensor technology [Hagemaier], [Birx], [Johnson]. Clyde Kizer, of the Airline Transport Association summarized the beliefs of many professionals involved in NDI that a method is needed for looking (inspecting) at an aircraft without reliability problems due to boredom while maximizing aircraft total inspection time during the life of the aircraft. In order to mitigate the above aircraft inspection problem. The goal of our Office of Naval Research sponsored SBIR program was the specification of a mobile robotic NDI system which would be reliable, user friendly, adaptive and relatively inexpensive. An essential component of such a system is a neural network based flaw detector. Our approach to specifying a robotic NDI system for aircraft maintenance was to exploit the previous Navy sponsored robotic deriveter developed by the Southwest Research Institute. This system was enhanced by adding a completely automatic recognition system given that neural networks could perform the recognition tasks with high reliability.

We decided to focus our investigation on eddy-current based sensors [Birx] since this aircraft inspection method is universally accepted and the tediousness of this manual inspection method invites automation aids. However, in addition to eddy-current base inspection we gathered information on other inspection methods such as magnetic-optic sensors [Shih], Ultra-sonic sensors [Prabhu] and x-ray radiography [Berner]. See [Brown] for an overview of NDI Sensors. In fact, Neural networks were applied with great success by Prabhu and his colleagues at NASA Langley to aircraft lap-joint disbond ultra-sound based detection. In late 1990 Birx considered that state of the art in automatic NDI using technologies such as infinite element analysis, inversion methods was still deficient in enabling reliable automatic inspection.

Birx reported in his PhD thesis (University of Dayton, 1990) significant progress in applying neural networks using complex (i.e. real and imaginary) neurons and weights. Eddy-current data

is usually provided as two components, I and Q, which are In-Phase or Quadrature-Phase with respect to the excitation current. Birx provides simulation results of a neural network which performs complex arithmetic on the I and Q components of EC data. He reports successful backpropagation training on fewer samples with complex arithmetic than were required for the same data treating both I and Q as real numbers. However, we found that savings in training cycles may not compensate for the additional computer time needed for complex multiplication. One complex multiplication takes more than twice the computer time of two real multiplications. Birx showed that the real part of the error does not change the imaginary part of the weights, and vice versa.

Our approach considered that total robotic inspection problem: automatic flaw detection and automatic measurement. Concerning robotic measurement, we investigated the modifications required to the Navy's deriveter and a new fast manipulator path planning technique [Tarokh]. The latter methods will be valuable in applying robotic inspection to aircraft wings and tails and when scaffolding is in place around an aircraft.

2.0 Eddy Current and Robotic Considerations for Aircraft Inspection

Eddy current (EC) testing is based on the establishing electrical current flow in the part to be inspected and detecting defects based on associated changes in the current flow pattern. A probe using an excitation coil energized with alternating current flow is placed above the surface of the part, and the alternating magnetic field produced by the probe induces the flow of electrical eddy currents in the part. Sensing is performed either by monitoring the impedance of the coil, which is dependent on the eddy current flow characteristics, or by measuring the voltage induced in a separate, passive sensing coil placed near the excitation coil. Flaw detection using EC techniques generally requires that the probe be scanned over the surface of the part to be inspected. Certain characteristic changes in the eddy current response are then associated with the presence of a flaw.

Although EC testing is very sensitive to the presence of flaws, the response is also affected by other factors which can interfere with flaw detection. These factors include changes in the geometry of the part, changes in the electrical and magnetic characteristics of the part, and variations in liftoff (the spacing between the probe and the part).

Nondestructive inspection of aircraft skin for cracks (primarily those cracks originating from fastener holes) can involve the inspection of the first and/or second layers of skin material. The most difficult is detection of cracks in the second layer, especially where the first layer is relatively thick or where there are variations in the thickness of the first or second layer. The first layer acts to significantly reduce the signal from the cracks and variations in thickness (e.g., from structural features such as ribs and spans) introduce artifacts in the data that must be distinguished from the signal features caused by cracks. For second layer inspection, the most likely required scan motion would be to center a probe over the fastener and rotate the probe so that the eddy current detector travels along the circumference of the rivet head. Automating this inspection process will require implementation of a procedure for accurately locating the center of the rivet head and then positioning a mechanism for circular scanning over the center. For

both first and second layer inspection, the presence of a fastener can also produce undesired signals; if the probe moves from the skin to the fastener, a large signal is generated from the fastener/skin interface. Also, if the fastener is a different material from the skin (e.g. aluminum skin, steel fastener), an even larger response is obtained.

The scanning requirements for EC inspection of aircraft skin are highly dependent on the layer to be inspected and the flaw size detection requirements. The greatest sensitivity to small flaws in both the first and second layers can be achieved by centering a probe with respect to the center of the fastener, and then rotating the probe around the fastener. This process greatly reduces the unwanted signal caused by the fastener/skin interface and by different fastener material since the distance from the probe to the fastener remains constant throughout the scan. The difficulty with this approach is that it can be time-consuming to precisely position the probe, and thus, the inspection time for a large number of fasteners can be considerable.

An alternate approach is to scan the probe in a raster pattern over the fastener and adjacent area. This approach has the advantages of simplicity and significantly reduced inspection time because the precise positioning requirements are eliminated. The flaw detection sensitivity is reduced, but detection of both first and second layer flaws has been demonstrated with this approach. Figure 1 shows images generated from raster scan data obtained from a wing section of an Air Force T-38 aircraft. The wing skin is aluminum alloy, the fastener is steel, the simulated cracks (0.375 and 0.2 inch long saw cuts) are radially oriented in the second layer, and the second layer has an edge oriented parallel to the row of fasteners. In both images, the presence of flaws are quite visible, even though the image also shows the presence of the second-layer edge.

The capability for detecting first layer flaws using a raster scan approach was demonstrated using a T-38 wing section similar to that described above. Figure 2 shows an image generated from a raster scan over a row of 8 fasteners. An indication from a first layer simulated crack (saw cut) which extended 0.05 inch past the edge of the fastener head is visible in the image. These data were from a very limited investigation where little optimization of the EC probe and test parameters was possible. It should be possible to obtain improved results with additional investigation.

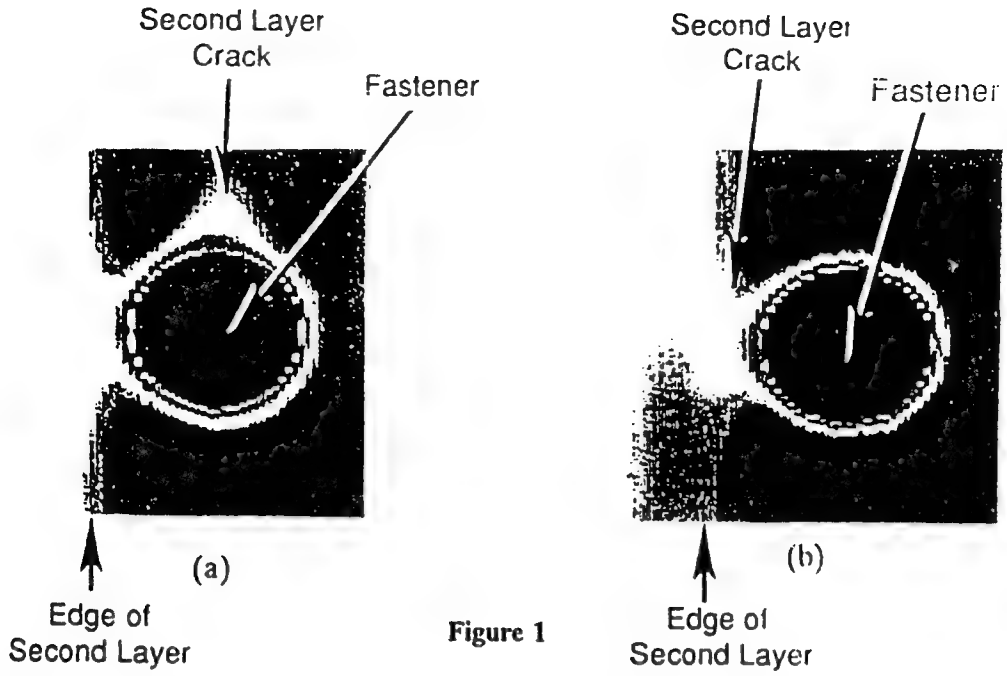


Figure 1

Eddy current images of second-layer flaws 9.53 mm (0.375 inch) long (a) and 5.08 mm (0.2 inch) long (b) in fastener holes. In both images, the faster is clearly visible, and the second-layer edge shows up as a vertical band on the left side of the image. The two-dimensional pattern of the flaw responses shows up easily against the large response from the aircraft wing geometry.

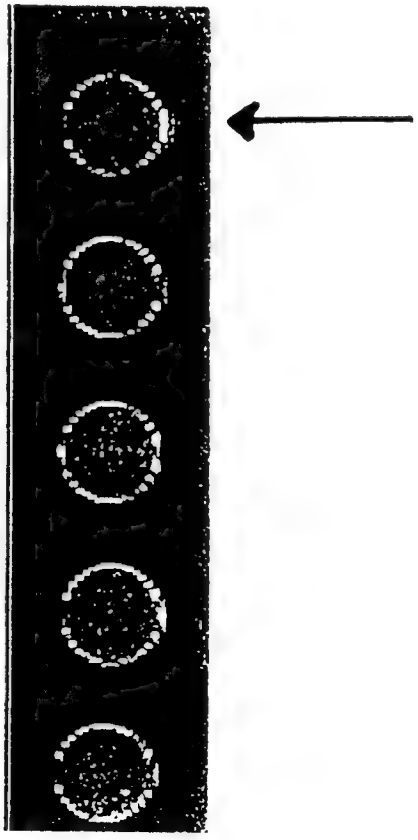


Figure 2

Eddy Current Image of First Layer Flaw Extending 0.05 Inch Beyond Fastener Head) In Fastener Hole of T-38 Aircraft Wing.

Based on the above results and the reduced requirements for a raster scan, it is recommended that a raster scan approach be considered as the primary choice for inspection of large areas of aircraft skin, especially if only a first layer inspection is to be performed.

2.1 Scanning Requirements

The limited first layer experiment described in the previous section was used to obtain preliminary data to determine raster scan requirements for inspection of aircraft skin. Based on this experiment, it was shown that flaws of a reasonable size could be detected using a raster scan having the following characteristics:

Scan track spacing	0.04 inch
Spacing between sampled data points	0.02 inch
Probe liftoff from skin surface	0.003 inch
Scan speed	0.5 inch/second

Any system for automated eddy current inspection of aircraft panels must provide a means for positioning and moving the EC probe over the area to be scanned. When the emphasis is the detection of cracks emanating from rivet holes in the skin, the system should concentrate scanning and analysis on the area immediately surrounding rivets and bypass other the remaining surface material. In order to do this it will be necessary to know the locations of the rivets relative to the coordinate system of the mechanism moving the eddy current probe.

Based on the prior experience at SWRI, we do not believe that it is feasible to pre-program the locations of rivets for an aircraft part and expect this information to apply to all (or even most) copies of that part. Many aircraft (particularly those that are nearing the limits of their designed life) were assembled manually and rivet spacings and placement along spars and stringers were put in "by eye". Even where the original positioning on a part was precise, manufacturing tolerances for joining subassemblies and stresses induced during thousands of hours of flight have caused enough distortion in the airframe that preprogrammed fastener location data will not match the existing patterns.

SWRI's Robotic Deriveter System was developed for the Navy and uses a machine vision system (IRI P256) with a camera and illumination source mounted on the robot end effector to determine the locations of rivets and other fasteners. This technique used a small field-of-view (approximately 1" X 1") to provide accurate ($\pm$.005") rivet position measurement for drilling. In order to accomplish this the paint had to be removed from the aircraft surface and lighting conditions had to be carefully controlled. The aircraft surfaces are often marked by paint flecks, scratches, dents, and other features that degrade image quality. Because of this, the computer vision algorithms used were relatively slow, requiring several seconds to locate and identify each fastener. Computers for image processing operations have improved greatly since the time the deriveter system was designed and the location problem could be performed much more rapidly using currently available hardware.

For automated inspection, high accuracy is not required so long as the scan pattern of the probe

covers the rivet locations. It is more important that the guidance system quickly determine the approximate locations of all rivets over a large area. Ideally this should be done without requiring the removal of paint or other surface coatings such as decals or stencils. In order to investigate this approach several tests were performed using machine vision techniques to locate rivets in painted aircraft panels. In most cases the pattern of the rivet head can be seen through the layers of paint as a characteristic circular indention in the surface. When the rivet was installed flush to the surface with no discernable gap, it was very difficult to locate rivets covered by several coats of paint. Best results were obtained when the light source and the camera were both at moderate angles (30° to 45°) from the surface so that the contrast of surface features was emphasized.

Small variations in surface flatness can be emphasized by an optical technique employing retro-reflective illumination. This technique, developed and patented by Diffracto Ltd. under the name "D-sight", utilizes a camera and closely located light source directed toward a reflective surface at a shallow angle. Light from the source is reflected from the surface under examination to a sheet of retro-reflective material where is directed back along toward the surface and then to the camera. Best results are obtained with large, flat, polished, reflective surfaces. In some cases a film of oil or other material can be applied to the surface to increase the reflectance. When this technique is used on highly reflective surfaces, surface patterns on the order of 0.001 inch can be clearly seen.

The D-sight technique was discussed with Omer Hagemaier's of Diffracto and its use in locating rivets under several layers of paint was investigated in brief laboratory tests. A machine vision system was used to observe samples of riveted aircraft material with several layers of paint. Best results were obtained with the light source and camera oriented at approximately 45 degrees to the surface with a sheet of retro-reflective material placed opposite the area being imaged. The position and orientation of the retro-reflective sheet was not critical and wide variations in position and orientation did not cause any discernable changes in the images. The reflectance of the surface being examined is critical; it was necessary to spread a light coat of oil over the painted wing material in order for the technique to be effective. When this was done, rivets that were very difficult to see under normal lighting were readily visible using the D-sight technique. Figure 3 shows the machine vision image of an area of a painted wing where the rivets on the right side of the figure are barely discernable. Figure 4 shows the same area under D-sight illumination clearly revealing the rivets and also showing evidence of slight surface deformations and skin buckling due to rivet installation.

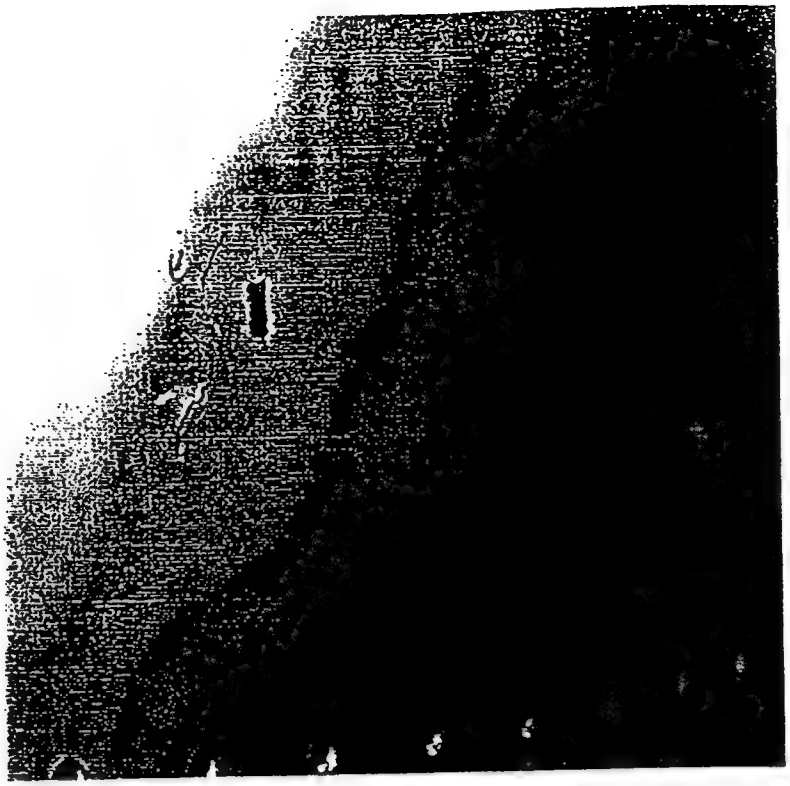


Figure 3

Machine Vision Image of Painted Over Rivets

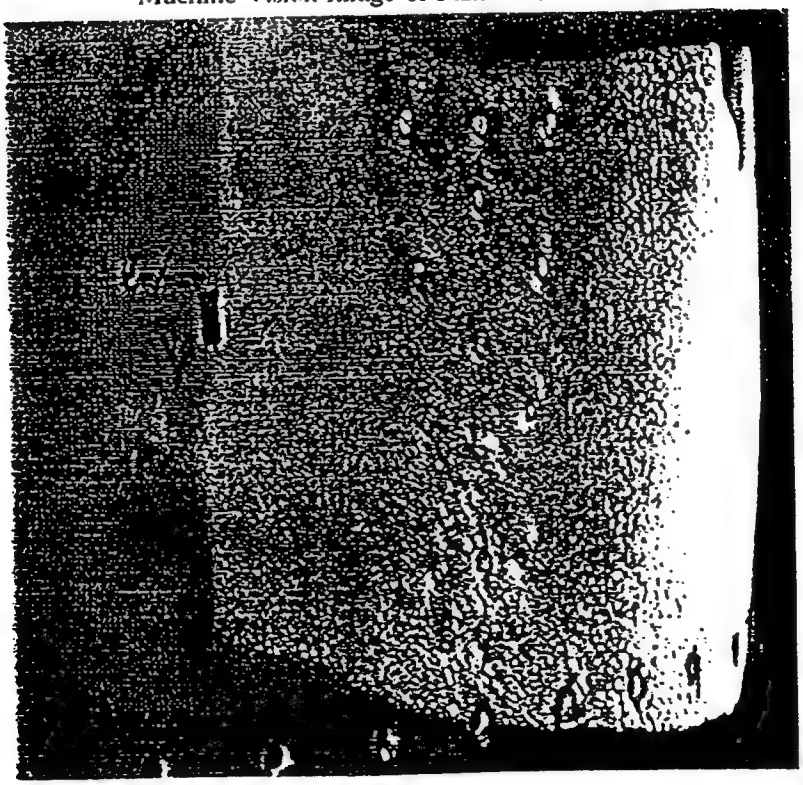


Figure 4

D-Sight Image of Aircraft Panel Showing Rivets Under Paint

2.2 The U.S. Navy Robotic Deriveting System

The Robotic Deriveter System (RDS) is a one-of-a-kind mobile robotic system for automated removal of rivets and eddy current inspection of the rivet holes. It was built between 1983 and 1986 by Southwest Research Institute for the U.S. Navy.

The RDS was delivered to North Island Naval Air Rework Facility (now Naval Air Depot) in the spring of 1986 and final acceptance tests were performed. The initial requirement for the RDS and much of the system specification was based on a program for modifications to F-4 aircraft but by the time the RDS was delivered, this program had been completed. Attempts were made to apply the RDS to several other workloads including engine cowlings and F-14 engine nacelles but the anticipated workload in these areas did not justify the retooling that would have been required to access these parts. For the next two years, the RDS was used only occasionally at North Island NAD. In 1987, Ogden Air Logistics Center at Hill AFB in Ogden, Utah, requested the use of the RDS for training technicians who would operate an automated derivating system being constructed for the Air Force. The system was shipped to 00-ALC and used briefly in 1988.

For the past three years the Robotic Deriveter System has remained at Hill AFB but is not presently being used for any productive work. The loan agreement has expired and 00-ALC is ready to return the RDS to the Navy but there is no available workload at NI-NAD. As a consequence, the mobile robotic system is not being utilized at this time. With proper justification, the system could be utilized for other automation or inspection projects in the U.S. Navy.

2.3 RDS Configuration

The Robotic Deriveter System was designed to be self-contained except for the requirement of 480 VAC, three-phase electrical power. In order to achieve this self-sufficiency, it includes a variety of subsystems.

The vehicle was designed specifically to allow transport of the robot, to hold the robot in a stable working position with a work envelope amenable to aircraft parts, and to provide a base for the auxiliary systems such as the robot controller, computer, machine vision system, and the eddy current subsystem. The vehicle uses rear-wheel (forklift style) steering to allow access into confirmed spaces and an air bag suspension system that allows the frame of the vehicle to be lowered to the shop floor to ensure stability during operation.

The robot is a Milacron T3-776 Robot with electric drives. It has 6 degrees of freedom and can carry a 150 lb payload. The stated repeatability of the robot is .02 inches but greater precision is possible if consistent paths and loads are used.

Tooling for rivet removal and inspection of rivet holes is included in an end-effector mounted to the robot wrist plate. The end-effector includes a turret which can be translated vertically or rotated by two stepper motors. The turret includes tools mounted so that appropriate rotation and

translation can position tools including:

- a machine vision camera for locating and identifying fasteners
- a drill for removing rivet heads
- a punch to drive the rivet shanks from the holes
- an eddy current probe to scan the sides of the rivet holes for cracks

The only one of these tools that is of interest for the planned inspection project is the eddy current probe. This probe is a small (approx. 2 mm diameter) cylindrical device that can be inserted into rivet holes. Two stepping motors are used to rotate the probe about center and to drive the probe along a radius to vary the diameter of the scan pattern. In the proposed inspection application, a different probe and scan pattern will be used. The current EC scan technique is of interest primarily as an example of the ability of the robot to position the end-effector accurately and repeatability enough to allow use of eddy current scan techniques.

The robotic deriveter system also includes a Milicron Version 4 robot controller, a general purpose minicomputer, machine vision computer and an eddy current instrument. Supporting subsystems including an air compressor, hydraulic unit, and vacuum are mounted on the vehicle.

The system software is written in Fortran 77 and runs on a Hewlett-Packard A-700 minicomputer. A Winchester disk is used to store programs and rivet location databases for different aircraft parts. The HP computer system also includes an assortment analog and digital I/O lines to monitor the status of the system and the process and to control the system.

Programs for teaching the robotic system the locations of rivets on parts to be derivated are included in the RDS. Since the system is mobile and the robot coordinate reference position is not fixed, a touch-off method is used to define the location of a work-piece relative to the robot coordinate frame. Coordinates of the touch-off points used to establish the reference frame are stored with the database of rivet coordinates. Rivet locations are then taught by using the teach pendent to move the robot end effector to the desired locations and storing the coordinates as determined from the robot controller. These locations are then refined by using the machine vision system to locate the precise center of each rivet on the part.

In the automatic derivet mode, the operator provides the name of the rivet location database to be used and manually (using the teach pendent) moves the robot to the part reference points. This provides coordinate data for transforming the stored rivet locations to robot coordinates for the actual part location. The HP computer then proceeds to retrieve the location of each rivet, move the robot to that location, correct for perpendicularity and offset, refine the rivet position and remove the rivet by using drilling and punching. If desired, the eddy current probe can be inserted into the hole to check the interior material for cracks.

The RDS used a contact method for determining offset and orientation of the work-piece. This was a consequence of the need to overcome robot deflections caused by the large forces encountered during drilling. In order to provide the stability needed for drilling, the RDS end effector included a ring that was preloaded against the aircraft surface prior to drilling. Sensors were included to measure the offset and orientation of the end effector relative to the aircraft

surface from the position of the preload ring.

2.4 Adapting the Robotic Deriveting System for Aircraft Inspection

Based on the requirements for eddy current inspection described above, the Robotic Deriveter System was analyzed to determine how well each of the existing subassemblies met the needs of the automated EC inspection system. The subassemblies used for drilling and punching operations will not be needed. These include the hydraulic and vacuum pumps and reservoirs. Many of the modules of the end effector including the drill, punches, marker, and J-bolt installation tooling can be removed from the system thus lightening the end effector and improving both the speed and positioning accuracy of the robot.

The requirements for scan track spacing and probe liftoff are beyond the capability of large anthropometric robots designed in the early 1980's such as the Milacron 776 when used in a "free-hand" mode. In order to provide the accuracy needed for reliable crack detection, a mechanism providing either circular or raster motions and having sufficient compliance to follow surface contours is needed. Such a mechanism could replace one of the end effector modules that will be removed but a better alternative is to replace the entire existing end effector with a simpler, smaller, lighter end effector for manipulating the EC probe.

The simplest configuration for a scanning module will be to use roller contact with the surface to maintain both standoff and orientation. Three roller balls or three wheels that can swivel to allow scanning in all directions can be used for surfaces with moderate curvature such as wings and some parts of the fuselage. Compliance could be provided by a flexible coupling and loading against the surface by either mechanical springs or air cylinders. This would be an inexpensive and reliable approach, well suited for initial investigations and tests but it would not be practical for scanning in confined spaces such as the wing root areas at the edges of engine nacelles.

For complete inspection of an aircraft, including these critical areas, a non-contact alignment system would be required. This can be accomplished by use of sensors to measure stand-off and an active control mechanism to adjust the position of the EC probe. Several different types of sensors could be used but optical triangulation sensors such as those manufactured by Keyence will be most suitable for the ranges and accuracy needed. Three sensors will be needed in order to measure both the standoff and orientation of the surface. The three sensors will need to be directed toward a small area so that the system can be used for inspection of rivets along the leading and trailing edges of wings. Older robot controllers such as the Milacron Version 4 do not provide the capability of adjusting the path dynamically in response to sensor inputs. In order to maintain probe standoff and orientation, the inspection module will include three independent axis of motion (z, pitch and yaw) controlled in response to the three sensor signals. A dedicated microprocessor will calculate the axis motions from measurements made by the three position sensors. The required dynamic response of this control system will be relatively modest since robot speeds during EC scanning will be low (2 ips).

One obvious implication of the high resolution and accuracy required for scanning is that

scanning the entire surface of an airplane will simply take too long to be practical. Any useful system must include a method for determining the +wcation of rivets and other fasteners so that the EC probe can be positioned to scan the area immediately surrounding the fastener hole. The Robotic Deriveter System used a machine vision camera to obtain an image of the fastener and locate the center accurately (±.005") enough for drilling but this approach required stripping the paint from the aircraft. For a practical inspection system it will be necessary to determine the approximate position of the rivets without removing the aircraft paint. A completely automated system could make use of the "D-sight" technique as shown by tests in SWRI laboratories but there are two major disadvantages to this approach:

1. D-sight requires shallow camera angles (approximately 45°) and placement of a large sheet of retro-reflective material opposite the camera. In order make all motions automatically, a very large end effector will be needed which win limit access to confined spaces.

2. The requirement for a higwy reflective surface will probably mean that a light oil coat will have to be applied to all painted surfaces. If this is to be done automatically, additional complexity must be added to the end effector.

An alternative approach would to rely on the human operators to mark rivet rows using an infrared reflective paint or dye. One convenient technique will be to use a chalk line with IR reflective powdered material. Two operators could quickly stretch the line over rows of rivets and mark them so they would be clearly visible to a video camera with an IR filter. The end effector would be equipped with a camera and IR rich illumination system that could image the surface from a distance of several feet. A machine vision computer would calculate the coordinates of each of the marked fines and transmit this information to the robot controller for path scanning. Such a semi-automated procedure will be more cost effective during the early stages of system development than a totally automated system.

In the existing system, the rivet locations are taught by manually moving the robot end-effector using the teach pendent. These locations are then stored and later refined by the vision system to locate the center of each rivet accurately. This manual positioning typically takes hundreds of hours during system development and verification. The use of the vision system with marked lines described above coupled with a robot path planner can drastically reduce or eliminate the need for manual positioning. The path planner also ensures collision avoidance and prevents damage to the eddy current probe and to the airframe. The use of a robot path planner is particularly important for confined spaces such as wing root areas at the edges of engine nacelles.

The use of a fast robot path planner, development by NETROLOGIC, was investigated for robotic aircraft inspection. A PUMA 562 robot was employed for preliminary investigations. PUMA 562 is kinematically similar to Milacron T3-776 currently used in the Robotic Deriveter System. PUMA 562 is an articulate 6 degrees of freedom manipulator with 3 major joints at waist, shoulder and elbow and 3 minor joints for the wrist.

An efficient and fast path method was developed by decomposing the 3-dimensional space into a number of rectangloid slices. A method was devised to work directly with arm configurations

instead of individual joint angles. These two aspects, newly decomposition and configuration control, greatly speed up the path finding procedure and make it possible to perform real-time planning in moderately cluttered environment that is encountered on the airframe. The path planner uses bitmaps of the arm and obstacles which are read from a file. The file can be created using either prestored airframe shape or image obtained on-line from a vision system. A neural network was utilized to determine the optimal arm configuration for preventing unnecessary arm movements during path execution. The path planning algorithm consists of four basic steps: 1) map the workspace, 2) input the target location as determined by a vision system, 3) rotate the waist axis while using a neural network to change arm configuration to avoid obstacles, 4) when arm is in the plane closest to the target, use an inverse kinematic solution to move the end-effector to the exact target location. Several laboratory experiments were carried out using the PUMA robot to determine the effectiveness of the path planner. The obstacles consisted of boxes placed on the floor and also several structures hanging from the ceiling. In all cases the path planner determined and executed a collision free path very fast. The path planner can easily be adapted to Cincinnati Milacron T3-776, GMF S-500 or similar robots.

2.5 Summary of Requirements

If EC inspection is done by raster scanning, the speed requirements of the robot positioner are not critical. Rows of rivets will be scanned at low speed, limited by the capabilities of the EC analysis equipment. Path planning can be used to move from one row of rivets to the next without having to cross large distances of unriveted surface. Only a small percentage of the time will be spent moving the robot from one row of rivets to another so the speed during this time wing have little effect on the total task time.

An important requirement for the system is the development of a vision-based robot path planner. This would relieve the tedious and time consuming task of manual path teaching and positioning. This would require integration of a wide-angle camera vision system and a path planner. Our initial investigations and experimentation have indicated that it is possible to achieve a relatively fast and efficient path planning for robotic aircraft inspection.

Sensing requirements will include measurement of standoff distance with an accuracy of .005" and orientation with an accuracy of $\pm 2°$. Both standoff and orientation sensors need to be as compact as possible in order to allow operation in confined areas or on surfaces with significant curvature. Sensors for collision avoidance will also be required to prevent damage to the EC probe and to the airframe itself. This should be capable of detecting obstacles extending more than 0.25" from the surface for a distance of two inches from the edge of the scanner module. At a speed of two inches per second, this will provide one second in which to stop the robot if an obstacle is detected. Sensors for fastener location (for circular scanning) and path identification (for raster scanning) will also be needed. A video camera mounted on the end effector and a machine vision computer capable of 256 levels (monochrome) with resolution of at least 256 X 256 pixels will be needed.

2.6 RDS Modifications for Inspection

The requirements for an automated eddy current inspection system differ, to some extent, from the current configuration of the Robotic Deriveter System. Initial tests of the neural network crack detection technique could be done with only minor changes to the RDS in order to prove the feasibility of the concept. More extensive modifications could be done later in order to gain greater speed and automatic operation.

Use of the vehicle as a mount for the robot is a valuable asset for investigations, tests, and system development. Many of the questions concerning automation of EC inspection of complete aircraft can only be addressed by scanning of large parts and complete airframes. The existence of a mobile system will allow testing different methods and procedures with very little investment in hardware systems. The Cincinnati-Milicron T3-776 robot has sufficient reach to cover typical aircraft parts without moving the vehicle and its payload capacity is more than adequate for the EC sensor and vision camera.

Of the auxiliary systems, the hydraulic system is used for power steering and should be retained on the vehicle. The compressed air and the vacuum systems will not be needed if a connection to shop air can be used to raise the vehicle for travel. The compressor and vacuum blower can be removed if desired or they can be left in place since they will not interfere with operation of the other components. The tool changer, consisting of a rotating turret just in front of the drivers station, can also be removed if desired.

The turret used to position the different modules required for rivet removal will not be used for EC inspection. It can be removed and stored for future use. In place of the turret, a single module containing an array of eddy current sensors will be used. This module will be include rollers to maintain reference standoff and orientation from the surface and will have a compliant coupling to the robot so that it can conform to variations in the position and orientation of the workpiece. The machine vision camera will be removed from the separate module of turret and attached to the new eddy current sensor module.

The robot controller, machine vision computer, and the Hewlett-Packard minicomputer will be reprogrammed and used for controlling the inspection system. Reprogramming the robot controller will simply require using the teach pendant to provide updated robot paths. Changes in the minicomputer software will consist primarily of removing sections of the code that performed the rivet removal processes while retaining the sections for subsystem diagnostics and maintenance as well as the operator interface and robot motion control. The image processing algorithms will be modified to speed up their operation by reducing the resolution and accuracy of the location determined. In addition, the image processing computer will be programmed to identify and locate straight lines as might be drawn on the surface to indicate rows of rivets.

In addition to the modifications to the deriveter system it will be necessary to add some new capabilities. The most important will be the eddy current signal analysis electronics needed for each of the eddy current detectors. If circular scans with a single EC element are to be investigated it should be possible to use the Nortec instrument already present on the RDS vehicle. If a large probe using 10 to 20 EC sensors is used to simulate raster scanning, it will

be necessary to obtain additional EC instruments or develop a method for multiplexing the signals while traversing the surface at very low speeds.

The U.S. Navy Robotic Deriveting System, although not optimal for the NDE task, will provide the major components for a flexible and cost-effective test-bed for investigation and development. The greatest advantage can be obtained by use of a multistage development and test program. The first phase of testing would consist of design and fabrication of an EC probe optimized for detecting cracks under rivet heads.

This would be either a module for circular scanning or a linear array of closely spaced sensors for one-pass linear inspection. The probe would initially be tested on test panels containing artificial cracks under rivet heads using laboratory instrumentation. After the first phase of laboratory testing is completed, the second phase would consist of scanning real aircraft for cracks. This would be done using the Deriveter System, modified as described above. The RDS would provide an economical framework for testing and verifying the operator interface, rivet location algorithms, robot speed control, as well as the fundamental issues of neural network analysis for crack identification.

Finally, after the procedures and algorithms are developed and thoroughly tested, it will be desirable to make additional modifications to the Deriveter System in order to improve its operational effectiveness. Items to be considered include upgrading to a faster, higher resolution image processing computer, programming more efficient methods for locating the rivets to be inspected, and upgrading to a newer robot. The Milicron 776 was designed for heavy loads as required in the derivating scenario. The EC inspection task requires supporting only a probe module and would benefit from greater robot reach. It may be desirable to change the heavy duty 776 robot for a lighter robot with greater reach. One candidate is a GMFanuc S-500 robot designed for sealant application. This has a larger work envelope but only 20% the payload capacity of the 776. An additional advantage of the S-500 robot is that it lighter than the 776 and can be ceiling or wall mounted. This would allow placing the robot on a scissors-lift so that it can be raised to inspect parts of aircraft that might otherwise be inaccessible.

3.0 Neural Network Based Flaw Detection

The network paradigm used in this study for automatic rivet inspection is a three layer feed forward network trained with back propagation of error. A network input layer takes the incoming eddy-current representations as its activations, these activations are multiplied by a "weight" and sent through a squashing function then sent to each hidden unit where it is summed up with the results of all the other input units. This sum determines the activation for the hidden unit and the whole process is repeated sending activations of the hidden units to the output units. The network is fully connected between successive layers, but has no connections which skip layers or are connected to the origin layer.

3.1 Data Representation Method

Present nondestructive metal failure analysis using eddy current signals is performed with an operator visually analyzing plots of the quadrature value of the sensor against the in phase value. Operators look at graphs of the data as both a two dimensional plot using both quadrature and in phase values, and as a one dimensional plot where the values are plotted against rotational position and analyzed separately (See Figure 5). Our first attempt at neural network fault classification used features derived from the two dimensional graph. The graph was treated like an image and features such as the percentage of black pixels in eight horizontal rectangles which sub-divide the image, were extracted from the graph. The resulting set of sixty-two features was then fed as input to a feed forward network. The output was trained to 0.9 if the graph were from a rivet that contained a flaw and, the output teacher pattern was set to 0.1 if there were no flaws in the rivet being scanned. These features were then trained on both alone, and combined with the features from the graphs of the values scanned from the same rivet at different frequencies (for a total of 186 inputs). The networks trained on this data set were able to learn the training data very well but were unable to correctly classify unseen data above chance levels, classic symptoms of overtraining.

Nets can be overtrained if the input set allows the net to key on idiosyncratic features of the input pattern rather than the features which characterize the domain the training data is intended to represent. This is a problem when the number of inputs is comparatively large compared to the total number of training patterns. In order to overcome this problem, a new training data set was constructed which keyed on the one dimensional plots in which each rotational location is treated separately. This had the effect of simultaneously greatly reducing the number of inputs while increasing the number of training patterns available.

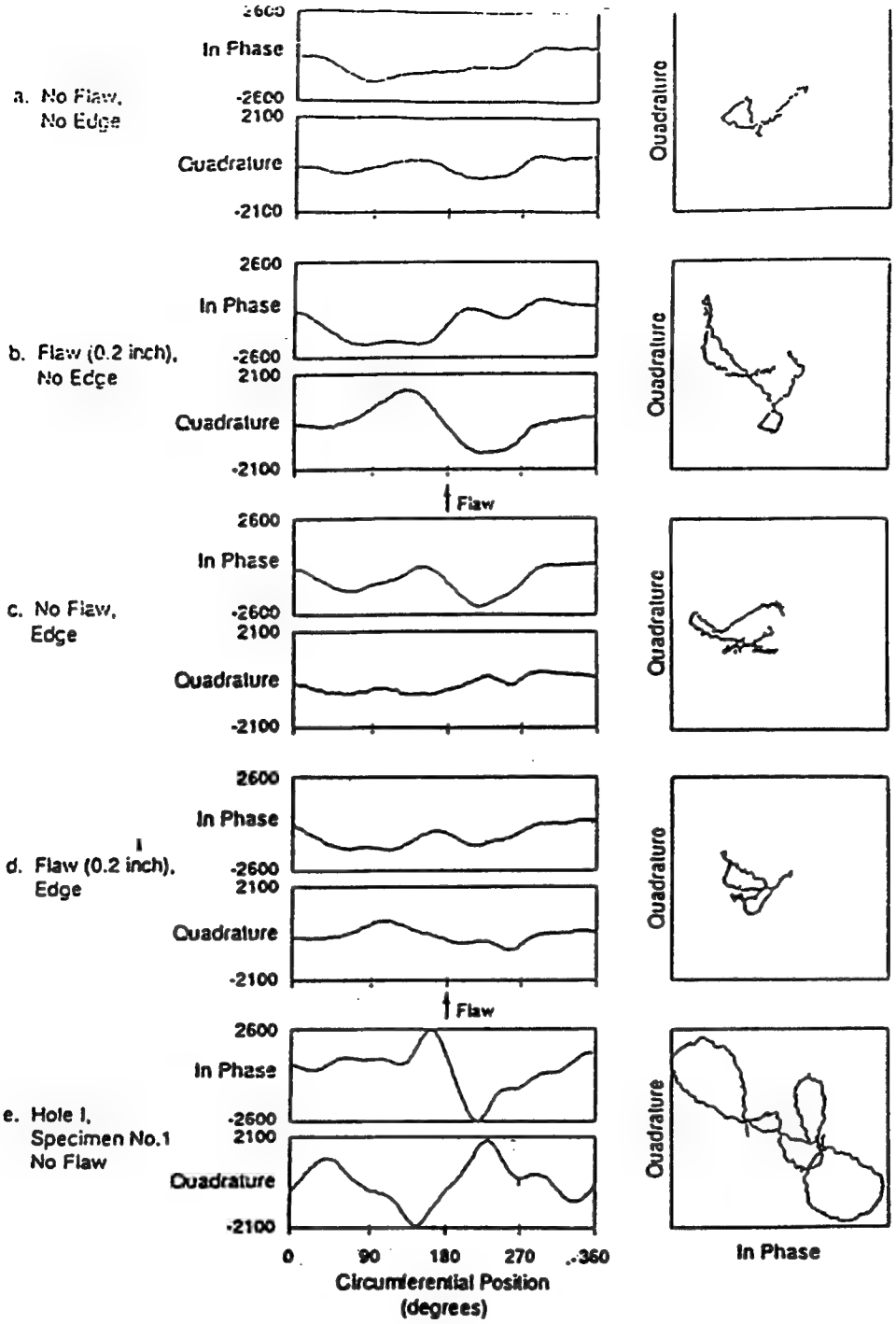


Figure 5

Flaw and Geometry Signals From a Simple Geometry Specimen (a-d) and Wing Specimen 1 Using Probe, C. Signal Levels are in μ volts.

3.2 Raw Data Set

The final neural network approach for rivet flaw recognition was performed on data from eddy current measurements taken from a probe at the low frequency of 1000 Hz, and at higher frequencies off 2000 and 3000 Hz. The experimental data were obtained by rotating the cross-axis probe 720 degrees around the target taking in phase and quadrature measurements at approximately every 0.7 degrees (See Figure 3.2). The targets were simple geometry specimens which have the same layer thickness as wing panels. Data was extracted

to determine the effect of the second layer edge and the change in thickness of the first layer across the scanning range. For application in the network data set, the in phase and quadrature measurements for all three frequencies were combined at each given angle measurement to form a single input set. The data was combined by taking the measurements at the different excitation frequency levels but at the same rotational position and putting them side by side in the same input pattern.

The order of inputs is not important in neural network applications as each input is processed simultaneously. The frequency levels were combined in order to present the network with all three levels in each input. The purpose of multiple excitation frequencies is to obtain signals which differentiate flaw and geometry feedback: since, low frequency data contain both flaw and geometry information while high frequency data is predominantly geometry information. The reason the frequencies carry different information is that the higher frequencies do not penetrate as well to the second layer while lower frequency probes do penetrate well and return a signal more dependent on the internal layer. By presenting the information in tandem, the network has the information it needs to separate the flaw component from its geometric counterpart.

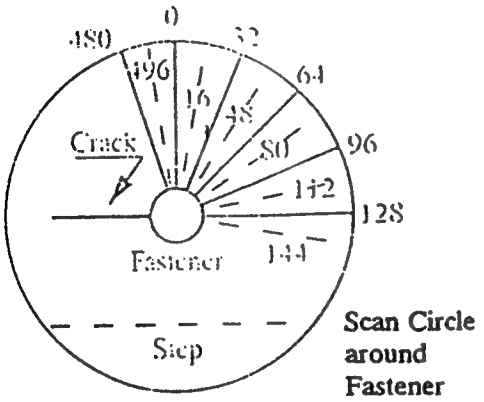


Figure 6
Scan Pattern Around Rivet

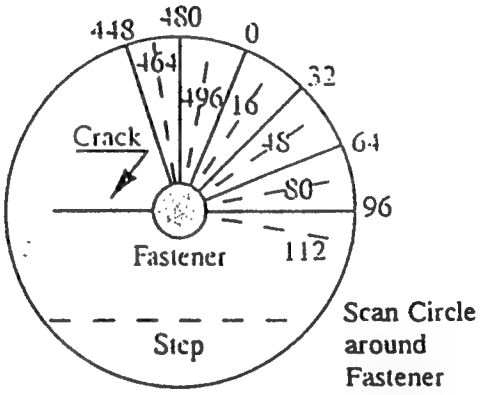


Figure 7
Offset data Reused for Training

3.3 Network Data Set

Each case has slightly over two full rotations around the rivet being measured yielding approximately 1024 raw data points, each with an in phase and quadrature component. Six

measurements correspond to each given rotational position, the two values at each of the three frequencies. These measurements occasionally had large quick value fluctuations (probably due to sensor anomalies). To counteract this, the raw values were smoothed with a low pass averaging function which spanned four degrees. Much of the information on the scanned rivet is carried in the change of sensor readings, so finding the change or first derivative of sensor values as a function of position is important in determining the presence of an irregularity. Instead of attempting to fit the signal to a function to enable derivation, the signal's rate of change or slope was calculated at each rotational position. A long range change value and a short range change value were added in with the smoothed raw value for a total of eighteen inputs to the network. Each of these inputs was normalized with its standard deviation to cover the range of approximately -1.5 to 1.5 The span of the slope features varied from 5 to 30 degrees. Optimal performance was found using a short range slope feature of 14 degrees and a long range slope feature of 28 degrees. The presence of both features was found necessary to prevent misfires while maintaining correct fires.

3.4 Network Configuration

The networks tested had from six to eighteen inputs, from five to ten hidden units, and one output unit, all were three layer feed forward networks trained with back propagation. Best generalization was obtained with eighteen inputs and as little as six hidden units. The minimum number of hidden units was chosen to reduce training time and force a more highly distributed problem solution among fewer hidden nodes. The network output patterns consist of only one output. This output is trained to 0.9 to indicate a fault sample or to 0.1 to indicate a nominal sample. An input set was considered to represent a fault if it was sufficiently close to the fault location on the bolt. A fault region must be determined because back propagation learning requires a teacher pattern; so the network needs to be presented with patterns indicating no fault, and patterns that represent a fault. Optimal generalization occurs when the training data corresponds to the natural transitions between decision regions.

In order for a network or any other method to differentiate between patterns, there must be some observable difference in the patterns. The dividing line for fault versus no fault training patterns should correspond to the actual change in input patterns. In order for a network to best identify the transition between fault and no fault data, points which characterize the central tendencies of a class of inputs must be presented along with points along the border of the decision regions. It is reasonable to assume that the location of the decision border varies with flaw size and orientation, but test runs indicate that the best range for declaring fault is those points within 20 degrees of the actual fault location. This resulted in the best network generalization. Generalization was determined by training the network on all of the data except for two cases which were excluded, then the network learning is turned off and the network is presented with the unseen pattern. The network's concentrations of fault outputs is then compared against the actual location of the rivet fault, if there is a fault.

3.5 Results

The feed forward network classified each sample on the scanner rotation as either fault or no fault, even rivets with flaws yielded no flaw samples because the data points beyond twenty degrees from the flaw location were trained as no fault. The network was trained on all available data, there were 18 cases with all three frequencies, and two cases that were held out to test the generalization of the network. The network was always able to learn the training data to within a maximum of 0.0113 RMS error. Different holdout cases were then tested to determine generalization to unseen data, there was always one holdout without a flaw and one holdout with a flaw to test any correctly classify at least some flaws. Each angle was presented separately, so for any given set of weights. Within the 20 degree fault range around a flaw,, the network always correctly classify at least some flaws. Each angle was presented separately, so for any given case there are many inputs and classifications. If each of these sets is viewed as an independent test, network performance varied across the holdout cases from 86% correct to 94% correct. This is the number of misfires plus the number of underfires divided by the total number of samples; a misfire occurs when the network classifies a fault on a sample that should be no fault, and an underfire occurs when the network classifies a fault input as no fault. A method for determining network performance that is more valid than viewing each input output grouping separately, is to rind whole regions where the net indicated fault presence, and then determine whether or not a fault actually exists there. For cases with flaws at least 0.187in deep, the number of correct fault classifications in a row was greater than the maximum number of incorrect firings in either the case of the faulty bolt away from the fault, or in the rivet that contained no flaw. The minimum number of contiguous correct firings was 37 while the maximum number of contiguous incorrect firings was 21. Even the holdout case with the smallest fault, 0.063in, fired 27 correct faults in a row although at one point it misfired 25 times in a row.

4.0 Fast Path Planning for Robot Manipulators

Efficient and fast path planning is achieved by a special manipulator and workspace representation, decomposing the 3-dimensional space into a number of 2-dimensional subspaces. A method is devised to work directly with arm postures (configuration) instead of dealing with individual joint positions. These two aspects, namely decomposition and posture control, greatly speed up the path finding procedure and make it possible to perform near real-time planning in a moderately cluttered environment. The path planner uses bit maps of the arm and obstacles, and a simple algorithm determines the sequence of configurations that the arm must take to effect a collision free path. The path planner has been implemented and tested on a PUMA manipulator in a moderately cluttered environment.

The general motion planning problem for a manipulator is to find a path from a given robot configuration to a goal position and orientation that avoids collision with a set of known obstacles. Several approaches to this problem have been proposed. Brooks [19831 has suggested a heuristic approach based on a separate free space description for the upper arm and the payload. His method, although reasonably general, is fairly complex. The idea of using artificial potential field for robot manipulator and obstacles was introduced by Khatib [1985]. In this approach, the

robot is pushed away by repulsive field of the obstacles and is drawn towards The goal by an attractive field. Koditscheck [1987], addressed the potential field control from a topological viewpoint, showing that an almost global solution for the navigation problem can be developed using an appropriately specified potential field function.

Another approach proposed by Lozano-Perez [1987] is based on the use of configuration space also called C-space, which is a space of dimensions equal to the degree of freedom of the manipulator. The idea is a build a space free of obstacles for successively more links starting with the first two links. Thus for an n-link manipulator, a two dimensional C-space is built for the first two links, then a three dimensional subset is constructed for the third link and this procedure is continued until an n-dimensional subset is built. This approach permits explicit characterization of the constraints on robot motion due to presence of obstacles. However, C-space obstacles tend to be complex especially when the dimensions are high. An improvement to this strategy is proposed by Gupta [1990], using a sequential search technique. This method results in one single dimensional plus n-1 two-dimensional path planning problems instead of one n-dimensional problem.

Another category of path planning approaches is hierarchical approximate cell decomposition introduced by Brooks and Lozano-Perez with subsequent contribution by other researchers [Faverjon 1986, Kambhampati and Davis 1986, Zhu and Latombe 1991). This approach consists of decomposing the configuration space of a manipulator into rectangloid cells at successive levels of approximation. Cells are classified as empty, full or mixed depending on whether they he entirely outside the obstacles, entirely inside the obstacles or neither. The planner searches a graph for sequence of adjacent empty cells connecting the initial configuration to the goal configuration. If no such sequence is found, it decomposes the mixed cells into smaller cells and searches again until either a solution to the path planning problem is found or until mixed cells are smaller than some predefined size. The problem of decomposing a mixed cell is to maximize the volumes of the empty and full cells resulting from the decomposition so as to find a path, or to conclude the absence of a path, as quickly as possible.

In spite of the considerable efforts, almost all existing path planning methods lack efficiency, require considerable computational power and time, and face major difficulties in implementation on practical manipulators. We believe that there are two main factors contributing to these problems. First the approaches tend to concentrate on a general but abstract manipulator-obstacles environment with little attention to common manipulator kinematic arrangement and geometry. Second, the path planning problem is generally set and solved at the joint level which requires finding a sequence for each manipulator joint angle to move the arm from the initial configuration to the goal configuration. In this report we develop a path planner on different premises. We fully take the advantage of the manipulator kinematic arrangement to simplify path planning. We also treat path planning as the problem of finding a sequence of manipulator configurations rather than a sequence of joint angles. Although the proposed method does not cover all types of manipulator, the class of manipulators that can be treated using the method is sufficiently large and includes most available industrial manipulators.

4.1 Decomposition of Manipulator Workspace

In this section we describe a method of decomposing the 3-dimensional workspace of a class of manipulators into a number of 2-dimensional workspaces. This will enable us to devise a very simple and efficient path planner.

The manipulators to be considered will have n_1 major joints (arm) and n_2 minor joints (hand). Usually $n_1 = n_2 = 3$, however, we allow the possibility of any number of joints. As with most other approaches, for the purpose of path planning, we use a simple conservative approximation for the minor joints and enclose the hand and any payload in a bounding box. We will assume that the for a particular joint angle 1, the motion of the major finks and the hand occur in a rectangloid sector of dimensions L x H x D, where L, H and D are length, height and thickness of the rectangloid sector. These dimensions are chosen such that for a specified joint 1 position Θ_1 and for all joint positions $\Theta_2,...,\Theta_n$, the arm from link 2 onwards lies entirely within the rectangloid sector, as shown in Figure 1. For reasons that will become clear shortly, we select the thickness of the rectangloid sector to be as small as possible. The thickness D depend on the thicknesses of links 2,..., *n* and any offset between these links. The thickness is small when there is no offset and the finks are thin. For a PUMA 560, L = H ≈2000 *mm* and *D ≈ 200 mm*, so that when Θ_2 and Θ_3 change through their entire ranges, links 2, 3 and the hand remain within a rectangloid of dimensions 2000 X 2000 X 200.

The assumption regarding the motion of links 2 onwards does not restrict the application of the path planner to specified manipulators. In fact many types of manipulators qualify the above description and include articulated (both elbow and parallelogram linkage X@X), spherical, cylindrical and Cartesian types. Examples of industrial manipulators of these types are PUMA 260, PUMA 560, PUMA 760, Cincinnati Milacron T^3 735 , Cincinnati Milacron T^3 886, GMF M-100 and Rhino XR-3, to mention a few. This kinematic arrangement enables us to develop a simple and efficient path planning algorithm, as will be seen. This kinematic arrangement enables us to develop a simple and efficient path planning algorithm, as will be seen. In order to simplify the presentation we will consider, from time to time, a robot with only three revolute major joints and refer to These joints as waist, shoulder and elbow joints. However, the method to be described is applicable to robots of the above type having any number of major revolute or prismatic joints.

Consider a rectangloid of dimensions L x H x D where *L, H and D* are length, height and thickness of the rectangloid. These dimensions are chosen such that for a specified joint 1 position Θ_1 and for all joint positions Θ_2, $\Theta_3,...,\Theta_{n1}$, the arm from link 2 onwards lies entirely within the rectangloid, as shown in Figure 2. For reason that will become clear shortly, we must select *L, H, D* to be as small as possible. The height H and length L can be selected to be equal to twice the sum of lengths of links 2,..., n_1 and the bounding box containing the hand and payload. This will ensure that the arm stays within the rectangloid when it is fully extended up, down, forward or backward. The thickness D depend on the thicknesses of links 2,..., n_1 and any offset between these finks. The thickness is small when there is no offset and the links are thin. For a PUMA 560, *L = H ≈ 2000 mm* and *D ≈ 200 mm*, so that when Θ_2 and Θ_3 change through their entire ranges, links 2, 3 and the hand remain within a rectangloid of dimensions 2000 X 2000 X 200.

Consider now a plane that divides the above rectangloid along its thickness into two equal rectangloids of dimensions $L \times H \times D/2$ and denote by R the region of the plane that is common between the two smaller rectangloids. Let us place the horizontal axis u and the vertical axis v on the above plane such that v is along link 1, as shown in Figure 3. The axis u is now perpendicular to link 1 but is parallel to links 2,..., n_1. The origin of the u-v axis is at the intersection of the plane with joint 2 axis. As the joint 1 position Θ_1 changes (the waist rotates), the plane and the rectangloid will also change position (rotate) so as to keep the arm within the rectangloid. The position (angle of rotation) of the plane ϕ is related to Θ_1 by $\phi = \Theta_1 + \alpha$, where α is a constant. It is to be noted that for a particular ϕ, the planar region R is uniquely specified in the 3-dimensional space. We will denote by R_ϕ the planar region corresponding to position ϕ. As Θ_1 changes from its minimum value to maximum value, ϕ alo changes through its range ($\phi_{max} - \phi_{min}$) and a volume that includes the robot work space is swept by the planar region R_ϕ. In doing so, R_ϕ also intersects obstacles that are in the robot work space.

Let the normal projection of links 2, ..., n_1 and the hand on the planar region R_ϕ be represented by the function $f_\phi(u_1,v_1)$ where for a point, $u=u_1$, $v=v_1$ located on the projection we have

$$f_\phi(u_1, v_1) = a_1, \neq 0 \qquad (1)$$

For the point (u_2, v_2) not on the projection, $f_\phi(u_2, v_2) = 0$. We generally set $a_1 = 1$, however, other values can also be used to assign weights to different parts of the arm. Note that when f $\phi(u_1, v_1) \neq 0$, we assume that the arm occupies the whole thickness D at the point (u_1, v_1). This is conservative when links have different thicknesses or offsets. Since the arm is located in a rectangioid of thickness $_D$ rather than the planar region R_ϕ, we grow each obstacle by $D/2$ on each side, as shown in Figure 4. This will allow us to treat collision in terms of intersection of the planar region R_ϕ and obstacles instead of the more difficult problem of determining the intersection of the rectangloid with obstacles. Let the intersection of R_ϕ with the set of obstacles $\Omega = \{\Omega_1, \Omega_2,..., \Omega_\mu\}$ be defined by the function $g_\phi(u,v)$, where μ is the number of obstacles. The point (u_1, v_1) belonging to the region R_ϕ intersects an obstacle if

$$g_\phi(u_1, v_1) = b_1 \neq 0 \qquad (2)$$

On the other hand if $g_\phi (u_2, v_2) = 0$ the point (u_2, v_2) belongs to a free space. In (2) b_1 represents the weighing of the point on the obstacle and we generally set it set to 1. Collision of the manipulator and obstacles occurs when a point in the region R_ϕ satisfies both (1) and (2). The intensity of collision is given by

$$e_\phi = \int_{R_\phi} f_\phi (u, v)\, g_\phi (u, v)\, du\, dv \qquad (3)$$

where the integral is evaluated over the region R_ϕ. It is more efficient and convenient to work with discrete intervals. Let the region R_ϕ be divided into a grid of m_1 X m_2 squares of side equal to $s = L/m_1 = H/m_2$. The arguments u and v in the functions f_ϕ (u,v) and g_ϕ(u,v) now takes only discrete values $u = 1,2,...,m_1$ and $v = 1,2,...,m_2$ If a's and b's in (1) and (2) can take

only binary values 1 or 0, the functions $f_\phi(u,v)$ and $g_\phi(u,v)$ represent bit maps of the arm and obstacles respectively. The intensity of collision is now

$$e_\phi = \sum_{R_\phi} f_\phi(u,v)\, g_\phi(u,v) \qquad (4)$$

where the sum is taken over all discrete values of u and v in the region. The accuracy of the representation can be increased by reducing the size of squares. Although any collision must be avoided, the notion of the intensity of collision is useful. Because of the conservative approximations, a low collision intensity value may in fact be a non-collision. Thus when e_ϕ is smaller that some predefined value, one may use a finer grid to determine more accurately the possibility of a collision. This is similar to the concept of cell decomposition []. However, we are now dealing with simple planar squares instead of cubes.

The position (angle) ϕ can also be discretized into N increments of

$$\delta\phi = \frac{\phi_{max} - \phi_{min}}{N} \qquad (5)$$

One way of determining N is to find the maximum number of non-overlapping rectangloids in the range (ϕ_{max} - ϕ_{min}. For a prismatic joint 1, $\delta\phi = D$ and $N = \phi_{max}$ - ϕ_{min}/D. For a revolute joint 1, suppose that when the arm is fully extended out its length is l. Let D the thickness of the rectangloid be the chord of a circle of radius 1, as shown in Figure 5. Then, $\delta\phi$ can be obtained from

$$D = l\sqrt{2(1 - \cos\delta\phi)} \qquad (6a)$$

or

$$\delta\phi = \cos^{-1}[1 - D^2/2l^2] \qquad (6b)$$

The number N of planar regions R_ϕ is now obtained using (5) and (6b). The set of discrete positions of the planar region is now $\phi = \{\phi_1, \phi_2,...,\phi_N\}$. It is to be noted that the number of planar regions can be made as large as desired by considering overlapping rectangloids. However, little will be gained by having a large N.

Even though for a given ϕ, the planar region $Ri \equiv R\phi_i$ is uniquely defined, the projection of arm, that is the bit map $f_i(u,v) = f\phi_i(u, v)$ is not unique. This is due to the fact that for a given ϕ_i, the arm can have infinite number of different postures*, if the joint positions vary continuously. However, if we allow only discrete changes, the number of arm postures will be finite. In the case of a manipulator with three major revolute joints (waist shoulder, elbow), the arm posture

* We use the term "arm posture" to denote the projection of links 2,..., n_l on the region Ri. Arm configuration will be used to specify both the projection of links 2,..., n_l on the planar region R_i ϕ_i, the angle of the planar region.

can change from arm fully extended up to arm fully extended down. For such an arm, intermediate postures correspond to different elbow up or elbow down arm positions. Figure 6 shows examples of arm configurations in the region R_i. The resolution of path planning algorithm depends on the number of arm postures V that we allow. As V is increased the resolution improves, however, more time will be needed to select a particular posture among the set $\mathbf{P} = \{P_1, P_2,...,P_v\}$. Working with arm postures instead of joint angles is crucial in our path planning method.

4.2 Path Planning Method

Consider a manipulator at some source (initial) configuration described by the planar region position and arm posture (ϕ_s, P_s). Suppose that the manipulator is required to move around the obstacles and place the hand at the goal position described by the Cartesian coordinates described by the vector n-dimensional vector x_g where $n \leq 6$. A set of manipulator joint positions corresponding to x_g can be found using an inverse kinematic method. Suppose that (ϕ_g, P_g) is the arm configuration corresponding to the Cartesian goal coordinates where x_g and P_g do not in general belong to the above defined sets $\phi = \{\phi_1,...,\phi_N\}$ and $P = \{P_1,...,P_v\}$. We can now select two arm configurations, say ϕ_i, P_j and ϕ_l, P_k that belong to the above set and are closest to $(\phi_s$ $P_s)$ and (ϕ_g, P_g) in the sense of minimum difference in the joint angle norm. The path planning problem is now to move the robot without collision with obstacles from its source configuration (ϕ_s, P_s) to the configuration (ϕ_i, P_j), then to the configuration (ϕ_i, P_j), and finally to the goal coordinates in Cartesian x_g. The first and last motion can be performed using an inverse kinematic control method, assuming that the obstacles are not very close to source or goal configurations. The major part of path is, however, from (ϕ_i, P_j) to (ϕ_l, P_k), and we now describe how this path can be planned.

The values of ϕ_i and ϕ_j, determine the direction of waist motion for minimum joint 1 angle change. In changing the joint 1 angle from ϕ_i to ϕ_j, the arm traverses several planar regions. The arm posture must also change from P_j to P_k, as the waist rotates. In order to determine the intermediate postures, we use a "joint space posture map." Each arm posture n the set P is uniquely represented by a point in the n-1-dimensional joint space. For the simplified PUMA arm mentioned earlier, the axis of the n-1 = 2 dimensional joint space are the shoulder and elbow joint values. The number of points, that is the number of arm postures, determine the resolution of the posture representation. In the ideal case of infinite number of postures and no obstacles, a straight line drawn in between the P_j to P_k would determine infinitely many postures that lie on the line and that the arm would have take to reach the goal. In our discrete setting, very few or no postures may be located on such a line. To resolve this situation, we use a cylinder of radius r in the n-1-dimensional joint space whose axis is the line connecting the postures P_j to P_k. The radius of r depends on the resolution, that is the number of arm postures. All postures within this cylinder are picked up for further processing. The selected postures are now sorted, are distributed among planar regions, and are checked for possible collision with obstacles.

Consider a typical configuration (ϕ_i, P_j), $\phi_i \in \phi$, $P_j \in P$, a set of obstacles $\Omega = \{\Omega_1, \Omega_2,...,\Omega_\mu\}$ and the planar region $R_i \equiv R_{\phi i}$. The next planar regions to be selected is either R_{i-1}, or R_{i+1} depending which one is closer to the desired plane R_l,. Let us assume, without loss of generality, that R_{i+1}

is the closer plane. Let the bit map of arm posture and obstacles on this plane be f_{i+1}, (u,v) and $g_i+1,(u,v)$, respectively and evaluate

$$e_{i+1} = \sum_{R_{i+1}} f_{i+1},(u,v)\ g_{i+1}(u,v)$$

Now if $e_{i+1} \neq 0$, there will be no collision and joint 1 position is changed (the waist is rotated), without changing the arm posture, so that the planar region is in the next location ϕ_{i+1}. However, if $e_{i+1} = 0$, we must select another arm posture among the set P and check for any collision. Note that in order to effect a collision free move from one plane to the adjacent plane an arm configuration must be clear in both planes. A physically meaningful criterion for posture selection is to pick one that is closest to the present posture P_j in the sense that a norm of joint position changes is minimum so as to reduce manipulator torque requirements. More precisely, let $\Theta^T_j = (\Theta_{j2},...,\Theta_{n1})^T$ be the vector of positions of joints 2,..., n_1 when the arm posture is P_j. Let the corresponding vector of arm posture P_k be Θ_k. Then the next arm posture P_k is chosen according to

$$P_k : P_k \in P \text{ and } \| \Theta_k - \Theta_j \| \qquad (10)$$

The arm postures are now sorted according to the distance from the current arm posture. The closest arm posture P_k is now selected and checked for collision using (4). If the closest arm posture is clear , it will considered as the new starting posture and a line is drawn between this new starting posture and the goal posture. This line now constitutes the axis of the new cylinder and the above procedure is repeated until either a suitable path is found or the absence of a collision free established. In the latter case, a higher resolution planner involving more planes and arm postures, i.e. higher N and V, can be attempted. Finally, there are several provisions for dealing with "dead-end" situations and for back tacking. These will not be discussed here due to space limitations.

It must be emphasized that, the determination of the optimum arm posture according to criterion (10) can be performed off line and the results can be stored in a look up table. In this case, we can evaluate $\| \Theta_k - \Theta_j \|$ for $j,k = 1,2, ..., v$ and arrange $\{P_1, P_2,...P_v\}$ in increasing order of the norm. This reduces the on-line time for the selection of next configuration to extremely small values.

4.3 Experimental Results

The path planning method described in this report was implemented on a PUMA 562 robot. The major joints 1,2 and 3, that is waist, shoulder and elbow, were used for path planning and the wrist was placed in a bounding box. For the purpose of path planning, the 320° waist joint angle was divided increments of 10° apart. This constituted 32 planar regions. Some 75 arm postures were defined. Thus the 320° planar regions and 76 postures determine various robot configurations and the robot is in one of these 2432 configurations for the purpose of collision detection. The obstacles consisted of, robot pedestal, several boxes placed on the floor and hanging from the ceiling, some in stalagtite-stalagmite fashion. Several experiments were carried

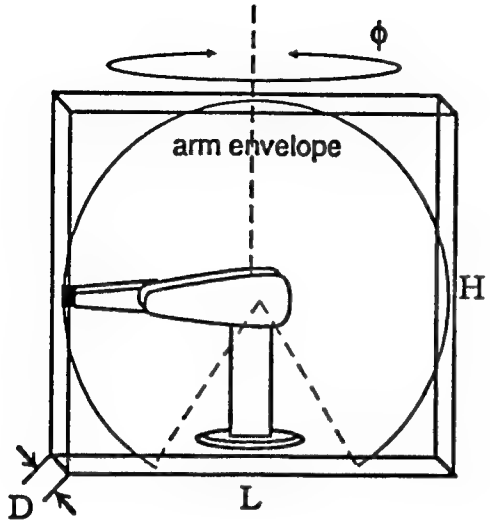


Figure 8. Rectangloid bounding the arm

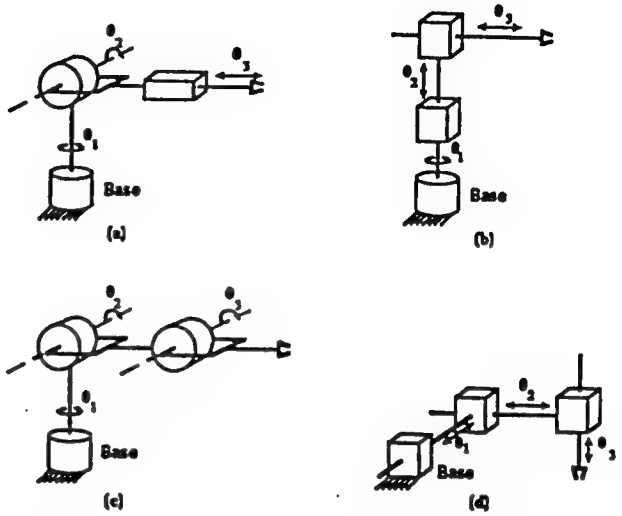


Figure 9. Various kinematic arrangements

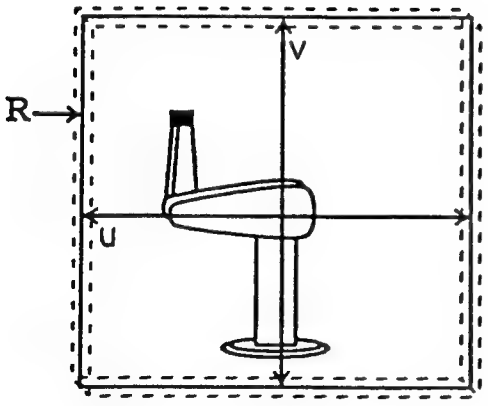


Figure 10. Plane R bisecting the rectangloid

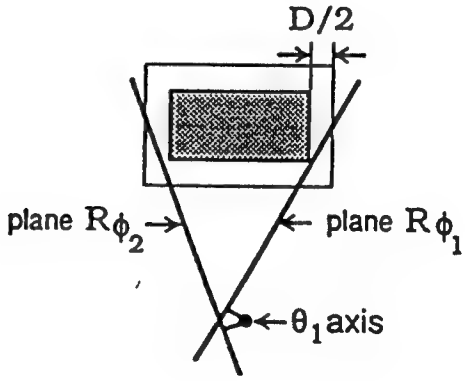


Figure 11. Overhead view of intersection of two planes with an obstacle increased by a radius of D/2

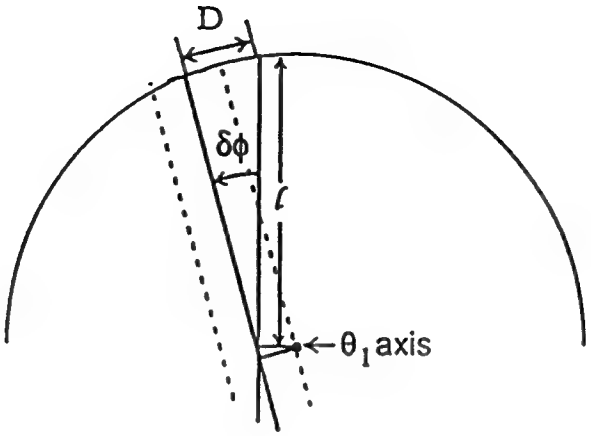


Figure 12 Incremental change in a plane angle

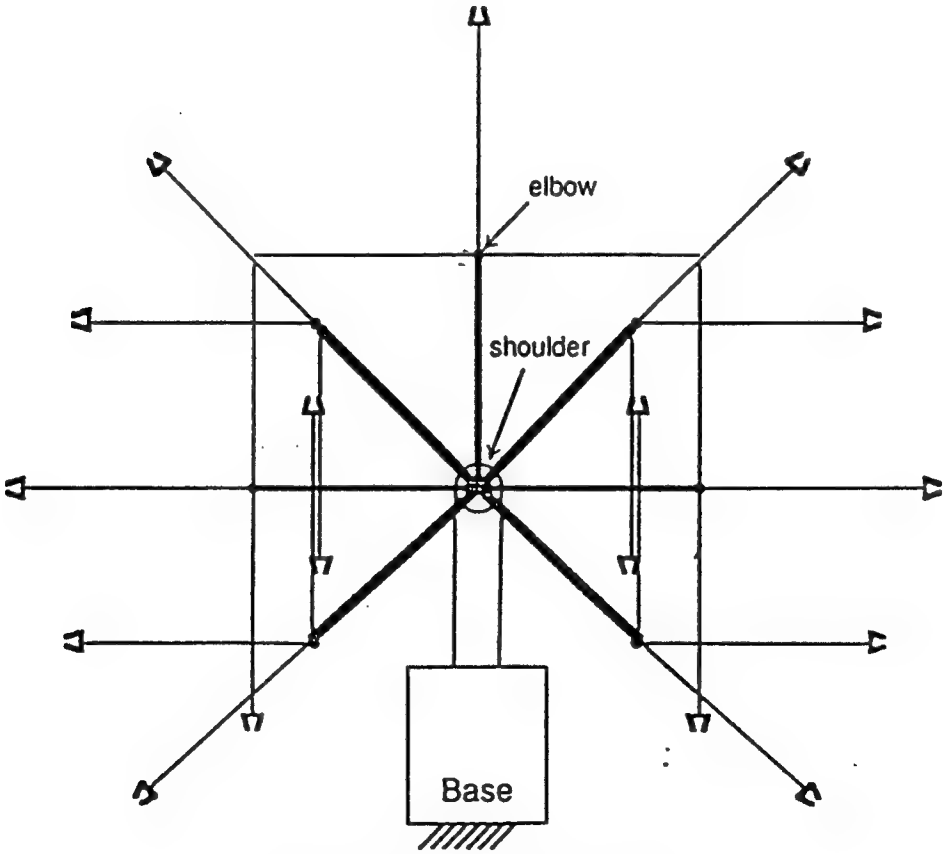


Figure 13 Arm postures for a plane R

out to determine the effectiveness and speed of the path planner. For an environment consisting some 15 obstacles, the path planner determined a collision free path free path extremely fast. The average time was about 2 seconds on a PC 486 machine. The software and various programs, their interactions, and modes of operations are described in a separate technical report [9].

5.0 Conclusions

A simple three layered feed forward net was found to be effective in fault detection, and performance should improve with fine tuning. This would include trying dynamic fault declare ranges to correspond with fault size, and different network outputs for different fault locations. The reason for the separate outputs is that the wing geometry may give faults at different rotational locations their own signature. Both of the alterations would be done in order to better arrange the training set to make fault classifications more normal.

A path planning method suitable for real-time implementation on a class of manipulators is developed and described in the report. The planner has a number of novel features. First, it decomposes the 3-dimensional search problem into a number of 2-dimensional problems. Second, to determine a collision free path, we have devised a method of working with arm postures, instead of dealing with individual joint positions. This simplifies path planning and greatly speeds up path finding procedure. As a result, we are able to perform extremely fast path planning in a moderately cluttered environment. Another major advantage of the planner is that vision information on obstacles can be directly integrated into the path planner. This is due to the fact that the path planner uses bit maps of the type which is directly available from a vision system and no additional signal processing is needed to convert vision information into a suitable form for the path planner.

Since we use conservative approximations, the path planner may not be able to find a path in a very cluttered environment, even when a collision free path does exist. When the environment is littered with small size obstacles, the number of defined arm configurations must be increased. This somewhat slows down the collision checking and path finding procedure. Our experience has shown, however, that even with several times the number of defined configurations, the path planner is still very fast (taking only several seconds on average). The path planner has been successfully implemented and tested on an industrial manipulator in a moderately structured environment. At present we are working on extending the path planner to deal with robots with redundant degrees of freedom, and to achieving motion planning for both positioning and orientation of the paylaod.

6.0 References

1. Berner, P. "NDT's Search for Tomorrow", Aviation Equipment Maintenance, September, 1991.

2. Birx, D. The Design of a Neural Network that Performs a Complex Mapping for

Phase-Sensitive Detection and Characterization of Eddy Current Impedance, PhD Dissertation, University of Dayton, December 1990.

3. R. Brooks, "Planning collision-free motions for pick and place operations," Int. J. Robotics Res., vol. 2, no. 4, pp. 19-44, 1983.

4. R. Brooks and T. Lozano-Perez, "A subdivision algorithm in configuration space for findpath with rotation," in Proc. 8th Int. Joint Conf. Artificial Intell., pp. 799-806, Karlsruhe, Germany, 1083.

5. A. Brown, "Seeing Beneath the Surface with NDE", Aerospace America, May 1992.

6. B. Faverjon. "Object level programming of industrial robots", Proc. IEEE Int. Conf. Robotics Automat., pp. 1406-1412, San Francisco, CA, 1986.

7. K. K. Gupta, "Fast collision avoidance for manipulator arms: A sequential search strategy," IEEE Trans. Robotics Automat., vol. 6, no.5, pp. 522-532, 1991.

8. Hagemaier, D. NDT Development in the Aircraft Industry, McDonnell Douglas Paper MDC 91KO003, Huntington Beach, CA and Westec 1991 paper.

9. T. Lozano-Perez, "A simple motion planning algorithm for general robot manipulators," IEEE J. Robotics Automat., vol. 3, no. 3, pp. 224-238, 1987.

10. S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots", IEEE J. Robotics Automat., vol. RA-2, no. 3, pp. 135-145, 1986.

11. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", Int. J. Robotics Res., vol. 5, no.1, pp. 90-98, 1986.

12. D. E. Koditscheck, "Exact robot navigation by means of potential functions: Some topological considerations," Proc. IEEE Conf. Robotics and Automation, pp. 1-6, 1987.

13. S. W. Magnetic-Optic/Eddy Current Imager, PRI Instrumentation Company Literature, Torrance, CA.

14. M. Tarokh, W. Adsit and L. Fountain, "Implementation of a fast path planner on a manipulator", Technical Report, Robotics and Intelligent Systems Laboratory, Department of Mathematical Sciences, February, 1992.

15. D. Zhu and J. C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," IEEE Trans. Robotics Automat., vol. 7, no. 1, pp. 9-20, 1991.

Multipath Signal Classification

James Johnson
Netrologic Inc.
5200 Springfield Pike, Suite 312
Dayton, OH 45431
(513)253-1558

1.0 Multipath Problem Description

An active radar guided missle uses a reflected return from the target for guidance. When emitted rays of the signal are reflected multiple times then a confusing situation results at the radar receiver similar to what happens when "ghosts" are received on a TV set. Both returns have the same PRF, the same RF frequency, very similar azimuth and elevation values and times of arrival, and similar signal to noise ratios. Since all of the commonly measured signal parameters about these signals is the same or similar, it becomes a real challenge to distinguish between the main signal and reflected signals, yet in order to get good missle accuracy it is vital to distinguish between the two signals. Our effort in this Phase I program was to establish a method by which the main signal and the multipath signals could be distinguished.

In a combat environment the multipath problem is compounded by the fact that many other emitters are operating in the same environment. Most modern EW receivers now are expected to operate in environments where the pulse density may exceed a million pulses per second. In this situation the receiver must separate the pulses into bins that correspond not only to the multipath, but also with the numerous emitters in the area. Now compound that with multiple missle launches and you have a most confusing arena in which to try to find a target.

Naturally when dealing with a problem of this type, the data becomes a difficult issue since you are dealing with data that is generated by rare and very expensive hardware. Fortunately, since Texas Instruments was our subcontractor they were able to supply data from an EW simulator and from actual missile tests where multipath was experienced. Using this data we developed a very sophisticated clustering algorithm and ran it on the data. It was able to cluster the data into clusters characteristic of various transmitters using data that was generated in the TI test facility. It was also able to cluster the data into separate clusters showing main return signal and multipath signals using the real missle data, and we were able to distinguish between what we believe were main path and multipath signals.

1.1 Radar Signal Data Characteristics

Our simulator generated data was collected in a table as a list of signals containing 6 features. The features were azimuth, elevation, signal-to-noise ratio, frequency, pulsewidth, and time of arrival. The objective of the clustering task was to use the information inherent in these features to recognize the similarity of the signals such that the signals could be identified as belonging

to a specific cluster having those features. If enough electronic intelligence is available, the type of radar having those characteristics can be identified. Hence the clustering would then actually identify the radar as being a specific type of radar. This type of information is vital to electronic warfare systems because the identity of the radar allows carefully constructed electronic countermeasures to be employed to defeat hostile radars. However, in our case we were not interested in ECM, but in radar signal processing. It was still necessary to discriminate among the returns to select the reflected radar signal from the missile from all the competing signals that were coming in. That means that the missile needs to know WHO is looking at it and WHERE they are in addition to its own target signal information. From the result of classification, the missile can ignore the allied radars, concentrate on processing its own signals and take care of enemy's facilities.

When the missile looks at the radar pulses being recorded by the radar receiver, it sees a confusing jumble of pulses all mixed together in a list. Since there is no bootstrap information about how to deal with this heterogeneous hodgepodge of signals, it needs an unsupervised classification algorithm to solve this problem; that is, the clustering algorithm must be able to generate clusters based solely on a knowledge of how radars operate and not on any specific *a priori* knowledge of what radars are present. Among these 6 features that can be passively collected from non-cooperative radars there are three, the signal-to-noise ratio, RF frequency of the emitted signal, and pulse width, which have good discriminant properties which we used in a first pass through the data to form initial clusters. Azimuth and elevation are two additional variables we saved for use on a second pass through the clustering algorithm in which the clusters formed from the first pass served as our bootstrap seed clusters.

The last parameter, arrival time, is not directly useful because it contains only a serial record of the time interval recorded between incoming pulses from the mixed signals. It is, however, a known fact that for many types of radar that the radar will send out signals with a fixed time interval between pulses. If there is only one radar sending signals to the missile, the difference between sequential signal arrival times for the whole signal set should be a constant. However, in the multipath situation we face a mixture of signals of the directly reflected signal plus the multiplely reflected signal from the same radar all mixed together in an unknown manner, and consequently the Pulse Repetition Frequency (PRF) can not be obtained directly from the original signal set. Therefore, there are only 5 features that can be used to classify signals from the original data.

In our proposal, an unsupervised neural network model, Brain-State-in-a-Box (BSB), was chosen as a key algorithm to classify multipath signal data. In the following sections of this report, we will discuss the failure of the multipath problem to satisfy BSB's input data requirements. Because BSB could not be satisfied by the nature of the problem data we created a new algorithm based on the K-Nearest-Neighbor (KNN) method to solve the problem. Our algorithm is called Hierarchical KNN (H-KNN).

2.0 The Unsuitability of the BSB Neural Network to this Problem

2.1 Coding Requirement in BSB Model

The BSB model can only process binary inputs. Therefore, any other input data formats need to be translated into binary code. For discrete cases which can be easily mapped into binary, such as Anderson's example of recognizing words made of 26 letters, the BSB works very well and demonstrates the capability of associative recall. Some commercial packages of neural networks like HNC's Anza Plus and NeuralWare's Explorer II use the same kind of example in their implementation of BSB. However, in the multipath problem, input signals are real numbers and belong to the continuous case.

If we use the BSB model to process continuous input signals, there are two fundamental requirements for coding the input data. One is that the order of the two representations should be homomorphic. To illustrate this requirement, let us assume the coding representation from real to binary is a mapping, C(x), where the C function behaves such that

$$\text{if } x < y, \text{ then } C(x) < C(y).$$

Now we know that the order of real numbers is defined by the difference of two numbers. It is calculated by $d = x - y$. If $d < 0$ then x is smaller than y. If $d > 0$ then x is greater than y. If $d = 0$ then x equals to y. That difference between real numbers is always used as distance along some analog axis of measurement. However, when you translate the continuous data into a binary you lose this property of analog numbers. The two representations are not homomorphic. For the binary codes, the difference between two numbers is defined by the Hamming distance since the discrete representation of the objects can not be located on the same axis to compare the order. For example if x=5 and y=1 then d=4. In binary the representation is x=101 and y=001. The Hamming distance between these two numbers is the difference between each of the individual columns of digits, i.e.

$$\begin{array}{l} x = 1\ 0\ 1 \\ \underline{y = 0\ 0\ 1} \\ d = 1+0+0 = 1 \end{array}$$

but d(real) is not equal to d(binary) even though the numbers they represent are the same.

If the continuous inputs are translated into binary codes for the BSB model, these binary inputs are considered as discrete objects by BSB and the model will distinguish them by their Hamming distance. As we saw from our example when we use the binary number representation of real numbers, the Hamming distance comparison does not keep the order of real numbers. So the inputs of multipath signals can not be simply written into binary numbers and fed into the BSB model. To solve this problem, we used a "closeness thermometer code" method to translate a continuous number representing a multipath signal into a kind of binary code. This coding technique is basically like a thermometer or regular analog measuring instrument. The data

indicator on this "binary thermometer" consists of a block of binary 1s (Figure 1). The position of the center of the block of 1s on the thermometer analog axis represents the real number. Using this representation a real number can be represented in binary and two different numbers can be compared. The two binary codes will have a Hamming distance which corresponds to the difference between their real values since the center of the block of 1s will be different for different real number representations. However, the maximal Hamming distance between any two data input codes is 2 times the block of 1s size. This means that the range of numbers which can be represented by this coding system is decided by the length of the blocks of 1s.

The second requirement for data representation for the BSB is accuracy. Using the above coding method, Drs. Anderson and Penz successfully demonstrated a solution to a simulated multipath problem by the BSB model [2]. However, the technique of using closeness coding in the BSB model with real world data did not work well at all. The reason is that the closeness coding has to represent real numbers instead of integers. In the case of representing real numbers, the accuracy of the representation of real numbers by a binary code is a big obstacle since the thermometer code does not retain the necessary accuracy.

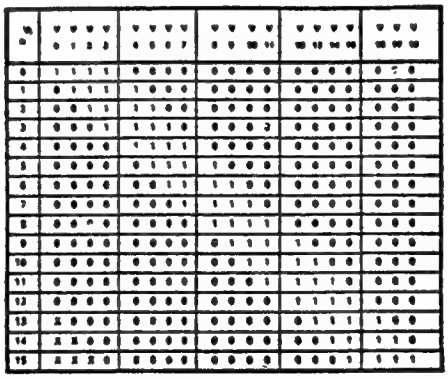


Figure 1. Example of Thermometer Code

2.2 Learning - Supervised Requirement of BSB

In Anderson's work he mentions that the BSB model has the capability of learning in an unsupervised manner. We found that this was true when there is a network that has already been trained on initial data. In the multipath problem however, the initial network has to be started from the untrained state where every weight is almost zero. Under these conditions, BSB has no idea which training pattern corresponds with which corner of the box in hyperspace and the updating of weights is therefore meaningless. Imagine that there is an initial network, with weights that are almost zero. For the unsupervised learning, use the training data to keep the iteration of the network going until the states of the network stabilize. What that state will be? It will be meaningless because the BSB model will select a random corner of the hypercube to represent an input state, but there is no measure of similarity to cause subsequent data to be

associated with the same state. Every data point will be represented by its own separate point in hyperspace and there will not be any measure of similarity developed. If you do not iterate the weights until they reach stability then the BSB has not really learned. The next data may cause the internal BSB representation to change and your previous data will have been lost. Since both of the two possible scenarios for unsupervised BSB learning are not workable, we conclude BSB unsupervised learning is not workable.

In order to use a BSB model it is necessary to start out with a supervised learning algorithm so that BSB could be used based upon the initial states derived from the supervised learning algorithm. In order to have a destination state for the input pattern in hyperspace, the BSB needs a target vector to start with. However, this is impossible to get directly from the signal information in the multipath problem.

2.3 Multipath Problem and BSB

Based on the discussion of section 2.1 and 2.2 we conclude the BSB model is not suitable to solve multipath radar signal problems. It requires that you know the answer before it can solve the problem. What then can be done to solve the problem? We postulated that a clustering algorithm could solve the problem if one could be devised to autonomously cluster the pulses into classes such that azimuth and elevation could be used to distinguish between multiplely reflected signals and directly reflected signals. Once these signals were accurately clustered we postulated that if the PRF of the missle transmitter was used and the times of arrival of the signals in the two clusters were used that one set of signals would be found to lag the other cluster by a small but constant delta time. The later signal would be the multipath signal.

As we look at the nature of the signals our receiver has intercepted we find that the multipath radar signals have no direct information about the nature of the signals, no identifying features about the radar and no indication as to how many radars are putting out those signals. This is information that must be derived from what can be gleaned from the jumble of signals received at the missile receiver. The requirement to cluster the radar signals springs from the fact that there is no direct data available to identify the radars and their locations. If the signals can be properly clustered, characteristics of the radars can be derived and the radars identified based upon the cluster characteristics and signal intelligence.

In our simulator data set, there are 6 individual signal features: azimuth, elevation, signal-to-noise ratio, frequency, pulse width, and time of arrival. From among these features, the location of radar relative to the receiver can be estimated from azimuth and elevation. Each incoming signal also gets a time of arrival tag assigned to it when it is processed by the radar receiver. The tag is a sequential time stamp given to each signal in the order it was received. From this, if you collect all the pulses from the same transmitter into a cluster, you can derive the pulse repetition frequency feature (PRF, the number of pulses received from that radar per second) from calculating the difference between the times of arrivals of the signals. If all the signals being received are emitted by one radar, this feature can be obtained easily by subtracting the time of arrival of the previous signal from the current signal. If, however, there is not one but an unknown number of radars, then the pulse trains mingle together in an unknown way and PRF

may not be directly computed. In fact, time of arrival if it is used as an initial computational feature gives no information about the classification of the input signals because we do not know what the time interval between subsequent pulses is without sorting them into clusters. Consequently our available feature set for initial clustering is reduced to five.

Of these five available features, not one can be used as the target feature to train the initial weights such that we might use the BSB model. As a result we have to look for a new algorithm.

3.0 KNN Algorithm

Intuitively, the K-nearest-neighbor classification algorithm is a good candidate for the multipath problem because it clusters data having similar characteristics. Having these clusters would give us hope that we could derive PRF and some delta times. However, we found that the KNN algorithm had limitations for this problem as well because KNN requires that you know something about the cluster centers before you begin to cluster the data. We give a brief description of the KNN algorithm here (the detailed definition and analysis can be seen in [4]) and go on to define an autonomous clustering algorithm from which we can derive the necessary signal information.

3.1 Algorithm Introduction

The k-nearest neighbor decision rule is defined by Fukunaga[6] as the following:

$$\hat{p}_N(x) = \frac{k-1}{N}\frac{1}{A(k,N,X)}$$

To classify a sample, consider an unknown sample X which is to be classified. The k-nearest neighbors of X are found among N samples, which consist of N_1 samples from w_1, and N_2 samples from w_2 which are stored in memory. Let k_1 and k_2 be the number of samples from w_1 and w_2, respectively, among these k-nearest neighbors. Then the formula (1) becomes

$$\hat{p}_{N_i}(X/w_i) = \frac{k_i-1}{N_i}\frac{1}{A}, \quad \text{where } i=1,2$$

Using the Bayes test for minimum error, we have

The algorithm needs to calculate the relation of the new sample to k previously classified samples and to place the new sample into the cluster nearest to the new sample.

For a modified k-nearest neighbor algorithm, there are k clusters represented by their centers.

$$k_1 > k_2 \rightarrow X \in w_1$$

$$k_1 < k_2 \rightarrow X \in w_2$$

A new sample will calculate k distances to these centers. Then the cluster which has the nearest center will include the new sample. This is expressed as

$$\text{distance}_i = | X - K_i |_n,$$

$$X \in w_j \text{ if } \text{distnace}_j = \min\{\text{distnace}_i\}$$

$$i = 1,2,\ldots,k$$

For KNN let there be n features in a given set of input data. If there are k known clusters, then calculate distances between each data element and the centers of every cluster. The cluster that is the shortest distance away, or the cluster center that is closest determines the cluster that the data is a member of.

3.2 Limitation of KNN on the Multipath Problem

For KNN the number of clusters has to be known. Furthermore, the center of each cluster has to be given. Otherwise, the distances can not be calculated and the nearest neighbor can not be chosen. For the multipath problem, these requirements translate to mean that the number of radars and the features of their signals have to be known before clustering can occur. These KNN prerequisites are, most irritatingly, just what we seek to derive from the signals, and these requirements, just as the BSB requirements previously, cannot be met.

If the KNN algorithm is adopted, it has to begin without any information about the clusters such as the total number of them and features of them. The only information it could have is what could be gleaned from the features of the signals such as the signal-to-noise ratio, pulse width, frequency and signal direction.

At this point we have exhausted all existing signal processing or clustering algorithms known to us and have not found any of them suitable for our cause. Therefore, we were compelled to invent a modified KNN algorithm suitable for multipath radar signal classification.

4.0 M-KNN, Multiparse KNN Algorithm

4.1 Algorithm Design

In M-KNN, which we developed specifically for the multipath problem, the set of 5 features; azimuth, elevation, signal-to-noise ratio, frequency, and pulse width; are used as the basic feature set. On this feature set we run a modified KNN algorithm to determine the probable cluster centers. Based on the rough clustering result, a validation pass is run on the data. This validation process uses the properties of the features mentioned before plus time of arrival (which we can now compute from information derived from knowing the rough clusters) to check every member in each cluster to verify that it is properly categorized as a member of that cluster.

In Figure 2, the blocks "classify" and "validation" are the heart of the algorithm. The logic proceeds as follows. The block labeled "classify" contains four underlying steps in the procedure:

1. Calculate distances between the current signal and the centers of existing clusters using Euclidean distance measurement;

2. Find the minimal distance, d_k, to any cluster. This is the distance to the center of cluster k which is the nearest neighbor of the current signal;

3. If d_k is smaller than a criterion usually described as the radius of the cluster, then the signal belongs to the cluster k;

4. Otherwise the signal is the center of a new cluster.

In the classification step where a signal is assigned to an appropriate cluster the criterion for the radius of the cluster is an important parameter that significantly affects the final clustering result. Since this needs to be discussed in detail we will devote section 4.2 (below) to a discussion of parameter analysis.

The validation check usually is not required in the KNN algorithm. In our algorithm, this step plays an important role by enhancing the accuracy of the clustering. The basic idea of the validation check is to find one feature of the signal that is an inherent property. In the multipath problem, the PRF satisfies this requirement. After a first pass through the data which determines a signal's probable classification, the legality of each member in a cluster can be verified by using this inherent property. Each signal which failed to meet the validation check is moved out of the cluster and put into a special class called the noise cluster. By this process, more accurate information can be gleaned to refine the clusters. The noise cluster can then be considered as a new set of unprocessed signals and fed into the algorithm again.

After the validation check the "organize" function will collect those signals which failed the validation check and ready them for the subsequent "classify" procedure. Therefore, the next classification step, using new clustering criterion, works only on those failed signals to generate

new clusters.

Using the rough clustering information derived by means of the information obtained from the first pass through the data, a second validation pass cycles through the candidate rough clusters and removes every illegal (based on a maximum radius from the cluster center) member of each cluster from that cluster and puts it into a "noise cluster". After a validation check on the modified clusters, all the clusters except the noise cluster are then considered definite clusters. We now can use the values of the azimuth and elevation characteristic of a cluster to define the location of a reflected signal or another emitter. The features of the cluster along with our computed PRF can be matched against intelligence data describing various radars to identify the type of radar detected. This completes the signal clustering of the data except for checking the clusters to insure they are not merely clusters of noise.

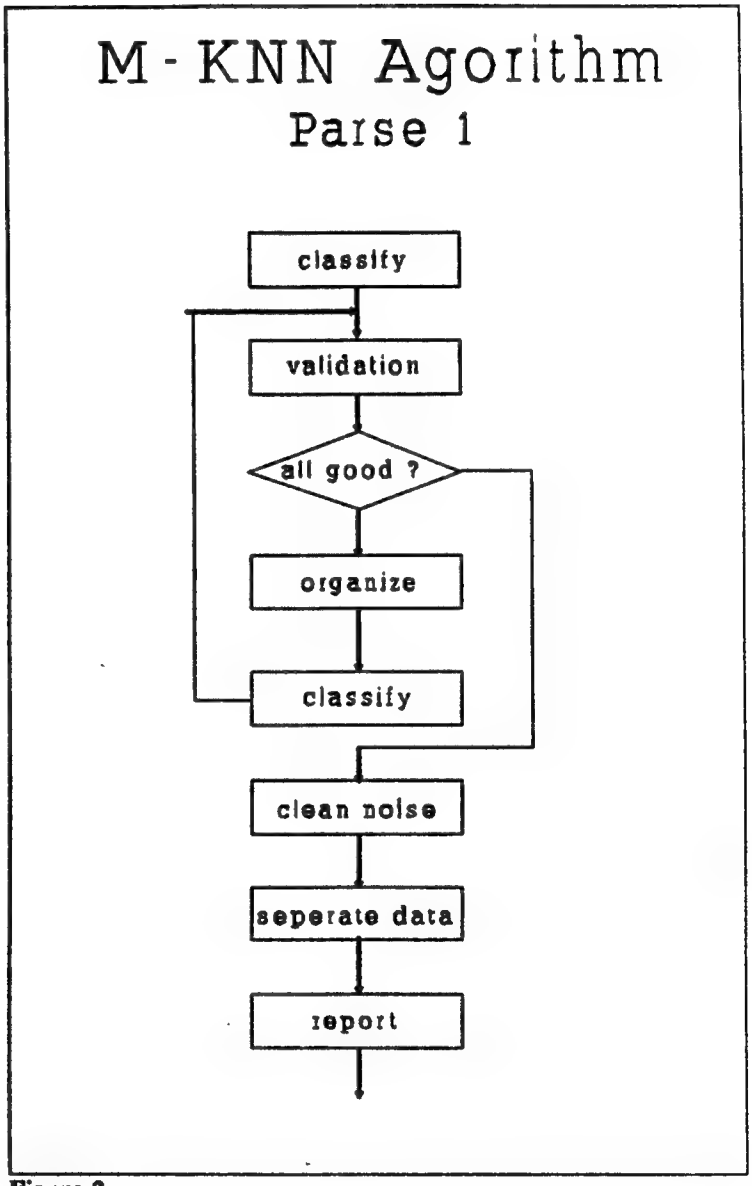


Figure 2

When every signal is validly clustered, the algorithm will check the legality of the clusters themselves. This is done by the logical process block called "clean noise". In this step, the algorithm only checks the size of the clusters. We selected a cluster size of 1% of the total number of signals as a minimum valid cluster size, and clusters with populations less than this were discarded. Actually, this type of cluster represents the overlap region of adjacent clusters. Since there is no valuable information which can be extracted from these clusters, the algorithm will remove these clusters from the data file. Cluster removal is done by the block "separate data".

A second pass through the clustering algorithm is now made using the data that was initially identified as noise in the validation process. The noise clusters which we originally threw away will be fed into the H-KNN algorithm again using a relaxed criterion for the cluster radius. A noisy value will be processed on this pass to find some target cluster which it is similar to. Otherwise it will be placed in a new cluster which will be evaluated at the end of the run to

determine if it has sufficient population to merit keeping it. Clusters with one or two pulses are not retained, but are discarded as being truly noise.

4.2 Analysis of the Algorithm

4.2.1 Automatic Adjustment of Centers

In the M-KNN algorithm, there is initially no information about clusters except the radius parameter, which is a desired pre-assigned value. Every time a signal is considered to represent a new cluster, the coordinates signal becomes the center of that cluster and its feature values become the feature values of the cluster. When a new member joins the cluster, the center will be adjusted by taking the mean values of the clustering value among all members as the new cluster center coordinates.

Using the above method to update the center of clusters, the algorithm cannot determine the center exactly. For instance, if the cluster takes in members on the border of the cluster the center will be pulled far away from its original position. This worst case happens only when the radius is larger than 1 and because of this we must limit the radius to values less than this. Fortunately this small radius requirement will be satisfied by data normalization which we discuss in the following section.

4.2.2 Normalization of the Input Signals

Generally, there are a wide range of magnitudes of features in a problem and this is certainly the case in the multipath signal problem. Here we find the range in pulse width is between (-2, 2) and the range of frequencies is between (7000, 20000). If we consider the distance of two signals on the absolute scale of features, the significant difference of pulse width will never affect the classification decision because on an absolute scale the changes in frequency will overwhelm the small changes occurring in pulse width. Therefore, to get these features where they will be equally noticed by a clustering algorithm it is necessary to normalize each feature such that the feature values will range from 0 (the smallest data value experienced) to 1 (the largest data value experienced).

In the multipath signal problem, there is no largest value which we can use as an upper limit to scale currently received values. Furthermore, because the processing is occurring in real time, the currently active data value or even a history of past values does not guarantee that a larger than the current maximum value or smaller than the current smallest value will not be received at some future time. Based on these limitations, the M-KNN algorithm normalizes the distance between the current signal and the center of the cluster. Also the algorithm uses the norm of the center of a cluster to be the scale in the normalization. In this way, the normalization is an on-line, real time normalization instead of the traditional sense of preprocessing the data.

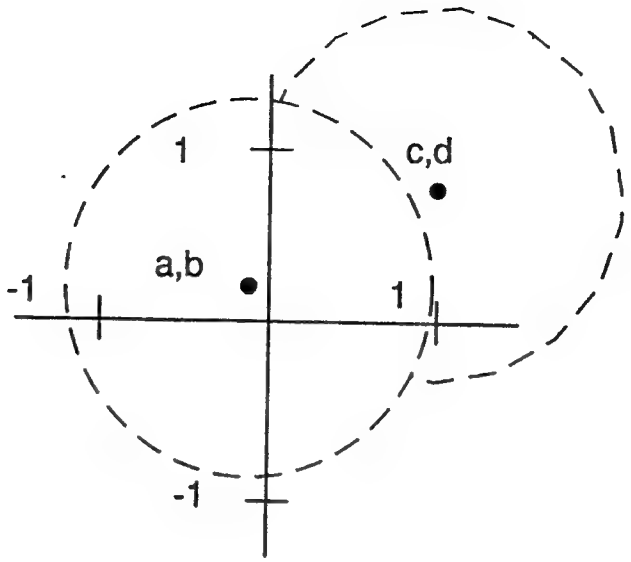


Figure 3. Example of grouping members in clusters during normalization process.

Now we must deal with the question of how, using the above normalization technique, the normalization process can keep every possible member grouped into the correct cluster. Let us look at the special case shown in Figure 3. There we note the center has coordinates (a,b) where a<<1, b<<1 and the cluster radius is r = 1. If there is one member close to the border of the cluster at coordinate (c,d) as shown in the figure, the membership test distance will be calculated as

$$\frac{|a-x|}{|x|} > 1, \quad \textit{and}$$

$$\frac{|b-y|}{|y|} > 1, \quad \textit{then}$$

$$d \gg 1 = r$$

Since this value exceeds the maximum permissible cluster radius, the member (x,y) will be lost to that cluster. Each feature on which we are clustering should keep the same order of magnitude for the all the range of values processed to avoid this problem. We note also that in the case we just analyzed the value (x,y) will not be lost to the system because (x,y) has such a different value from the cluster center that it will be incorporated into another cluster.

For the multipath problem, the first issue to be addressed is not to classify every single data value but to identify what the clusters locations must be. Even if there are a few data values which are initially considered as noise because they are closer to the border of the cluster, the majority of the signals will give correct information regarding the cluster centers, i.e. radar signal

locations, and knowing this then the algorithm can go on to do a perfect job by iterating through the data. Therefore, the M-KNN algorithm will give a good prediction of the cluster centers based on the distribution of the "very good" and hence very tightly clustered elements in the input data, instead of relying on the low noise requirements followed by other methods. This means that the information of the cluster (target radar) can be obtained from the center of cluster. If the distribution of the clusters is "good" with a tightly packed center then even if the algorithm loses some of its legitimate members on the edge of the cluster there are still enough members to give information based on the features of the center.

Now consider another case where the cluster has a "bad" distribution. In this case the number of signals classified into the cluster is small because the scatter of the cluster may exceed the critical radius value. This scatter will be initially recognized as noise in the M-KNN algorithm using the pre-defined cluster criteria, but will probably be picked up as a cluster on the second pass when the radius criterion is relaxed.

4.2.3 Choosing the Clustering Criterion

a. Radius of the clusters

We use the radius of the cluster as criterion 1 in the M-KNN algorithm. As noted in the discussion above, the radius can not be allowed to become too large compared to the scale of coordinates of the center, but the question is how do we determine the radius of the center before we even do any clustering? We can establish an upper limit on the radius size by computing a composite radius for all the features used for clustering the signal. The fact that we must compare features having different scales of magnitudes requires that we normalize the differences between the cluster centers and the data values for each feature separately as described above. Knowing that we are going to normalize the differences between the signals and the cluster centers allows us to compute the maximum value for a variable we call the composite radius. We can use this composite radius to estimate the maximum value for the individual cluster radii.

Let us now examine the worst case where every feature's value for a received signal fell on the edge of its cluster. In that case where each signal's value was on the edge of the radius, its maximum permissible radius value would be 1. It further follows then that the maximum for the composite radius is the condition where every signal lies on the border of its own individual feature cluster and therefore has its maximum permissible value of 1. It follows that the maximum permissible value for the composite radius would be equal to the sum of the maximum values of its components. Since these values are all one, the maximum value for the composite radius is equal to the number of features which are being clustered or N_f.

b. arrival tolerance criterion

The arrival tolerance criterion, the second cluster validation criterion, is used to determine the allowable tolerance for the difference of arrival times between two neighboring signals in the same cluster. An inherent property of radars is that their PRF is always a constant for a small time period. In normal operation many pulses are usually emitted by a transmitter which have

the same PRF. This large number of signals provides enough consistency in the received data to allow the M-KNN algorithm to cluster the data and characterize the signal such that with the proper signal intelligence the emitter can be identified. To compute the PRF the algorithm compares arrival times of the pulses in the rough clusters to determine the interval between pulses or the pulse repetition frequency. When the PRF is used as a validation check to substantiate a pulse's cluster membership, the criterion can be applied very stringently. In our test, this criterion is set at 0.5% of the possible time difference.

c. solid cluster criterion

The solid cluster criterion is the third cluster checking criterion in the algorithm. Even after every signal has passed the stringent requirements of the arrival tolerance check, the algorithm will always predict a lot of clusters. Most of these clusters have only 2 or 3 members as compared to the hundreds and thousands of original signals. We note that for these small clusters they either they have too few members to give accurate information, or else they include so much noise such that they are spuriously removed far removed from the clusters they actually should belong to. In both cases the algorithm considers these clusters as noise and ignores them when analyzing the target features.

In the second pass through the data if these noise sets are larger than a critical threshold, then the algorithm will apply relaxed radius criteria and will process this set again to extract some information to allow it to perhaps form new clusters based on information that was initially hidden in the noise set. In the next section, one example based on the simulation data is given to illustrate this process.

5.0 Radar Signal Clustering Computer Processing Results

5.1 Simulation Testing Data Results

5.1.1 Example of Data Set

In Table 1, there is an example of simulation data which we initially used to attack the problem of multipath radar signals. For the purpose of testing our clustering algorithm, this set of data gives the identification number of emitters for each signal, which is shown in the last column of the table. This information was not available to the clustering algorithm. There are 6 features which were used by the clustering algorithm: azimuth, elevation, signal-to-noise ratio, RF frequency, pulse width, and time of arrival. The total number of signals was 294 from 10 separate emitters. In the testing the M-KNN algorithm successfully identified these 10 emitters from these 294 signals without any preliminary information about them or any signal intelligence. Because the algorithm uses an on-line data normalization, the pre-processing of the original data set consisted of a pass to remove the last column containing the emitter ID. In both cases reported below, there are only 3 iterations needed to satisfy the validation check for the whole data set. One thing which needs to be pointed out is that the total number of signals available

from each emitter was very different. For instance, there are more than 140 signals from the emitter 2, but only 3 from emitter 9.

5.1.2 Clustering Results Using Different Sets of Criteria

For the simulation data set, two sets of tests were run using the M-KNN algorithm. The only difference between these two tests is that different sets of criteria were chosen.

In the test 1, criterion 1 was chosen as 0.15 and criterion 2 was chosen as 0.3, and criterion 3 was chosen as 0.01. Because there are 294 signals, by criterion 3, one cluster will be considered as a noise cluster if it has only 2 members. By this group of parameters, when every signal passed the validation check, emitter 9 was still not identified. This case is shown in table 2. Also the number of clusters considered noise is still quite large after the first pass through the data (more than 30 noise clusters which is 10% of the total data). A second pass through the data was then run on the noisy clusters to attempt to reduce the numbers of clusters. Table 3 shows the result of this second pass through the data using M-KNN processing, and on this pass emitter 9 was identified successfully. Using the criteria in test 1 on the simulation data set, we show it needs two parses of M-KNN processing to identify all 10 given emitters.

In the second test on the simulation data set, criterion 1 was 0.7, criterion 2 was 0.5, and criterion 3 was 0.01. The results of this test are shown in Table 4 in which 10 emitters were successfully identified during one pass through the clustering algorithm. We were able to set the parameters to successfully cluster in one pass by our knowledge of how many emitters were present and the experience we gained from running test 1. This is a special case incorporating that knowledge, but test 1 remains a more general case which did not rely on this information. Because of the additional information we have we were able to increase the maximum radius of the clusters to 0.7. However, by the analysis we did in the preceding sections, the value should be 0.5 or less. This indicates that our radius criteria may be overly stringent for this case, but we will leave it to future projects to determine criteria needed to relax the radius. Actually, since these two testing results give the same prediction about target information it increases our confidence that we are in fact validly clustering on key components of the signals.

5.1.3 Analysis of Results

The results of our clustering algorithm are shown in a series of plots. In Plot 1, the horizontal axis is azimuth, and the vertical axis is elevation. The positions of each dot represent the position of signal emitters. In the picture, there are 9 clusters of signals which can be obviously distinguished. Emitters 2 and 6 (identification number given by the original data set) are very close and it is hard to tell based solely on azimuth and elevation if these are two separate emitters or if there is simply a noise effect. When we look at other features such as signal-to-noise ratio, pulse width, and frequency then it becomes very evident that emitters 2 and 6 are very different. In fact, these two can easily be distinguished as two types of radars and will not be mixed. Using the results obtained from M-KNN clustering, the type of radars can be identified if a matching-feature-list with a corresponding radar type is supplied to the algorithm

from intelligence data. Therefore, we demonstrated with this example that the position and type of radars can be recognized using the M-KNN algorithm without resorting to any pre-existing information regarding the presence or absence of any particular radar.

As mentioned before, the M-KNN algorithm throws away a certain amount of noise signals based upon criterion 3. Practically speaking, signals in a noise cluster may be actual radar pulses, but the number of signals detected is too small to be reliably clustered. These kinds of trashed radar signals can be considered to be an unimportant existing radar. We may call them unimportant because in the real situation, if a radar is sending only a few signals (e.g. emitter 9) which the missile receiver is detecting, it means that this radar is not paying any attention to the missile yet. Before the radar detects the missile, the radar will require more signal return information from the missile than it is obviously getting at this point. Emitter 2 in the data set is an example of an emitter which finds the missile an item of interest and this is obvious in the number of pulses it is directing toward the missile in an effort to characterize the missile and track its flight. Under these conditions the large number of signals from the emitter provides more than enough information for the M-KNN algorithm to identify and locate the radar.

5.2 Physical Principle Involved in Multipath

When dealing with multipath there is a physical principle which will help us to determine which signal is the multipath signal. This principle is based upon the fact that radar travels at a constant speed in the atmosphere. This means that if two packets of energy are emitted at the same time from the missile transmitter and one takes longer to get back to the receiver than the other, then it is clear that one has traveled further. This fact is used by radars to compute ranges to the target. We also know that a radar signal that is reflected more than once, i.e. a multipath signal, travels a longer distance than a directly reflected signal and will take longer to return to the missile. We can use also use this fact to identify the multipath signal. Because the signal ray geometry of the multipath and directly reflected signals will be changing slowly relative to radar traveling at the speed of light, we can determine a constant delta time between the direct path signal and the multipath signal for a second or two time interval. By finding this delta time and noting the relation among pulse clusters we can determine that the one arriving first is the directly reflected signal and the one arriving later is the multiplely reflected signal. Knowing the delta time would allow us to discriminate against the later signal. Since our algorithm is capable of solving for the delta time between the direct and reflected signals, we can solve the multipath problem. The next section describes how the delta time between clusters is obtained.

5.3 Computation of Delta Time Between Direct and Multipath Signal

Using clusters 4 and 5 in Figure 4 as an example we proceeded one step further in our clustering process and reclustered the signals based upon our computed PRF and the 2 values for maximum cluster radius (criterion 1) and PRF variation (criterion 2) (Tables showing this are not included here due to space limitations). Once we had clustered the data we were able to compute the delta time between the PRF clusters. A frequently appearing number in the delta time column was 3.0 or 2.8 or 4.0 or some multiple of these. We believe these represent the time delay for multi-path

signals. There are many larger numbers in the delta T column as well. We attribute these to missed pulses and poor clustering.

6.0 Conclusion

It is apparent that our algorithm is not perfect, but there is undeniably an element of promise in the technique. We could do better if we had better data to cluster on, and if we had good ground truth data. We were able to obtain delta times between clusters which were of an order of magnitude to lead us to believe we had isolated the time difference between the direct and multipath signals. We clearly have a promising approach to the basic problem of discriminating between the two multipath signals. What remains is the task of cleaning up the algorithm and making it real time.

7.0 References

[1] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, *Distinctive features, categorical perception, and probability learning: some applications of a neural model,* Psychological Review 84: pp.413-451.

[2] P. A. Penz etc, *Radar Signal Categorization Using a Neural Network,* proceedings of IEEE, Vol.78, No.10, October, 1990

[3] P. A. Penz, *The closeness code: an integer to binary vector transformation suitable for neural network algorithms,* proceedings of IEEE neural networks, San Diego, 1987

[4] I. Thomek, *A generalization of K-NN Rule,* IEEE Trans. on SMC, SMC-6, pp. 121-126, 1976.

[5] HNC, Inc, *Brain State in a Box,* chapter 15, "User manual of Anza Plus", 1989.

[6] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* Academic Press, New York, 1972.

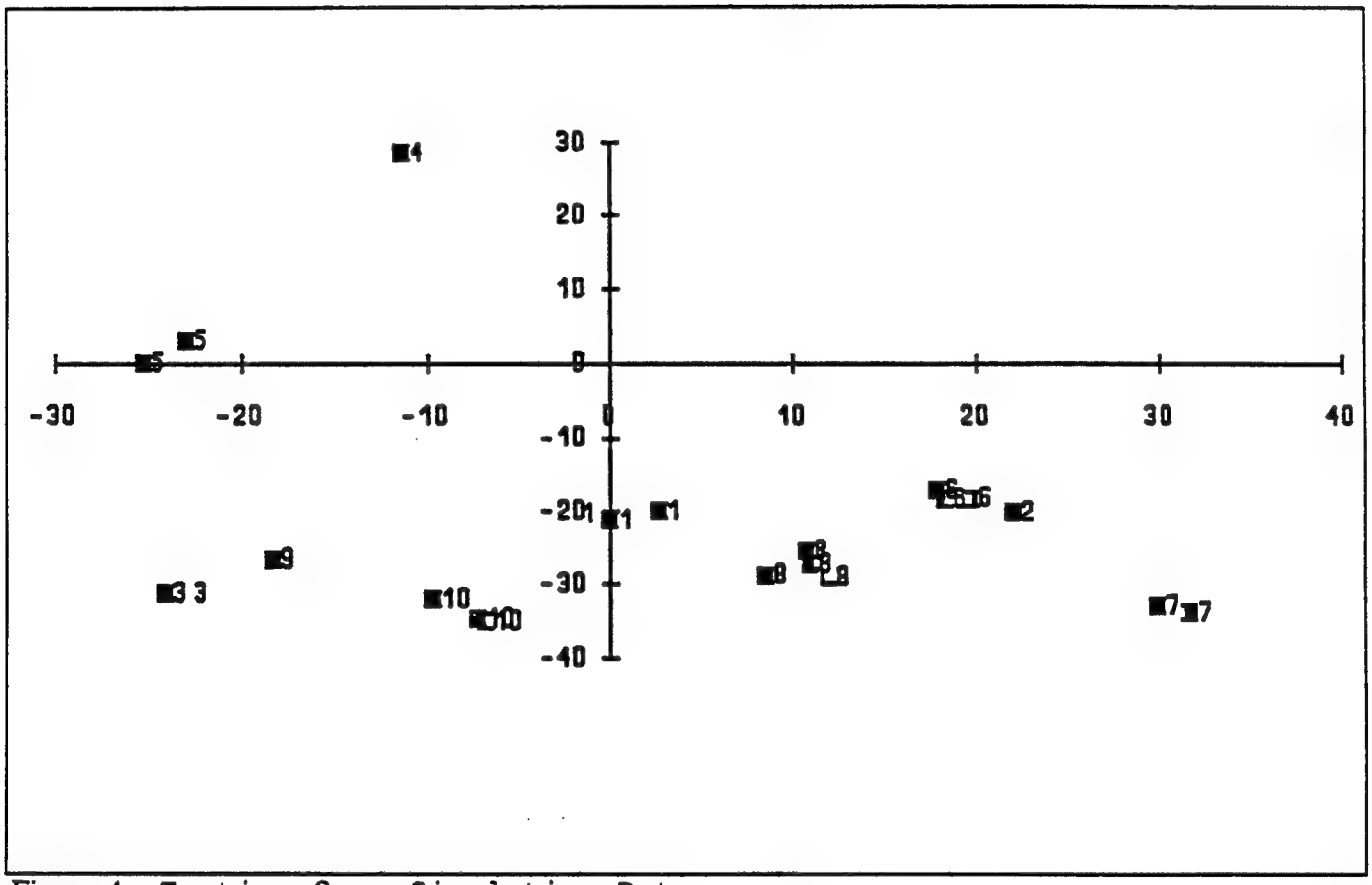


Figure 4. Testing 2 on Simulation Data

Table 1. Simulation Data of Multipath Signals

from Dr. A.Penz

Azimuth	*Elevation*	*SN_ratio*	*Freq*	*P.W.*	*A.time*	*Emitter*	*Order*
21	-20	-109	9066	2.186	1	2	1
34	-35	-98	9437	1.489	12	7	2
-12	27	-81	9214	0.399	20	4	3
-6	-34	-85	10054	0.421	53	10	4
-26	0	-86	9210	0.397	58	5	5
23	-17	-108	9030	2.191	75	2	6
16	-16	-97	9342	1.399	97	6	7
21	-22	-108	9015	2.195	112	2	8
-25	-30	-83	9023	0.416	117	3	9
19	-21	-109	9032	2.195	149	2	10
8	-29	-83	9805	7.156	164	8	11
20	-17	-109	9018	2.21	186	2	12
20	-19	-96	9335	1.402	213	6	13
23	-21	-108	9041	2.207	223	2	14
32	-30	-98	9435	1.375	251	7	15
24	-21	-108	9051	2.21	260	2	16
22	-20	-109	9011	2.194	297	2	17
19	-17	-97	9345	1.384	330	6	18
25	-20	-109	8997	2.185	334	2	19
23	-19	-109	9049	2.215	371	2	20
-12	30	-80	9224	0.398	374	4	21
21	-21	-109	9037	2.194	408	2	22
-25	4	-86	9090	0.401	428	5	23
9	-29	-82	9779	7.052	432	8	24
-7	-36	-85	10069	0.418	442	10	25
25	-19	-109	9063	2.186	445	2	26
3	-19	-72	8692	0.417	452	1	27
18	-18	-97	9345	1.421	454	6	28
26	-19	-109	9018	2.2	482	2	29
30	-34	-98	9345	1.402	502	7	30
-24	-32	-83	9026	0.422	505	3	31
22	-20	-109	9007	2.182	519	2	32
22	-20	-109	9013	2.202	556	2	33
21	-22	-97	9328	1.426	565	6	34
21	-20	-108	9062	2.196	593	2	35
22	-19	-109	9055	2.195	630	2	36
21	-21	-109	9032	2.192	667	2	37
21	-19	-97	9346	1.404	685	6	38
8	-29	-82	9888	7.108	702	8	39
21	-22	-108	9006	2.196	704	2	40
-15	27	-80	9267	0.399	728	4	41
21	-16	-108	9032	2.213	741	2	42
33	-35	-98	9423	1.317	751	7	43
23	-22	-108	9037	2.189	778	2	44
20	-18	-98	9342	1.412	804	6	45
-26	0	-85	9245	0.399	814	5	46

Table 2. Result of Testing 1 on Simulation Data

FAILURES = 54 (after 1st validation check)
FAILURES = 12 (after 2nd validation check)
FAILURES = 3 (after 3rd validation check)

The List of Identified emitters

Cluster	Member	Az	El	SNr	Freq	P-W	(Real Em.)
0	123	22.439	-19.951	-108.659	9031.722	2.204	(2)
1	7	29.857	-33.000	-98.000	9427.143	1.379	(7)
2	13	-11.308	28.538	-80.615	9230.308	0.400	(4)
3	4	-7.250	-35.000	-85.250	10074.000	0.420	(10)
4	4	-25.250	0.000	-85.500	9219.500	0.400	(5)
5	27	17.889	-17.259	-97.333	9339.813	1.399	(6)
6	9	-24.111	-31.333	-83.000	9042.333	0.405	(3)
7	4	8.500	-28.750	-82.250	9808.250	7.090	(8)
8	4	-23.500	4.000	-85.750	9177.750	0.398	(5)
9	3	2.667	-20.000	-72.000	8634.000	0.419	(1)
11	3	12.000	-29.000	-82.333	9782.333	6.909	(8)
13	15	19.000	-21.733	-108.600	9029.267	2.201	(2)
18	3	-9.667	-32.000	-85.667	10072.000	0.419	(10)
20	3	-23.333	2.000	-85.667	9293.000	0.401	(5)
28	9	18.444	-17.778	-96.889	9339.109	1.391	(6)
29	10	31.600	-33.900	-98.100	9426.400	1.403	(7)
30	6	-7.667	-34.000	-85.833	10066.333	0.419	(10)
31	4	10.750	-25.500	-82.000	9761.250	7.010	(8)
32	5	0.000	-21.200	-72.000	8708.600	0.419	(1)
33	3	-23.000	-31.333	-83.667	9057.000	0.402	(3)
37	3	18.333	-18.333	-97.333	9339.667	1.408	(6)

Table 3. Result of Testing 1 on Simulation Data

FAILURES = 1(after 1st validation check)

The List of Identified emitters

Cluster	Member	Az	El	SNr	Freq	P-W	PRF	(Em.)
0	3	20.000	-19.000	-97.000	9345.333	1.419	3370	(6)
1	4	11.750	-29.250	-82.500	9814.750	7.042	543	(8)
7	3	-18.333	-26.667	-97.667	10069.000	1.690	1276	(9)
13	3	-2.000	-20.333	-72.000	8647.000	0.418	830	(1)

Table 4. Result of Testing 2 on simulation Data

FAILURES = 56(after 1st validation check)
FAILURES = 18(after 2nd validation check)
FAILURES = 3(after 3rd validation check)

The List of Identified emitters

Cluster	Member	Az	El	SNr	Freq	P-W	PRF	(Em.)
0	140	22.021	-20.157	-108.657	9031.352	2.203	37	(2)
1	7	29.857	-33.000	-98.000	9427.143	1.379	337	(7)
2	13	-11.308	28.538	-80.615	9230.308	0.400	354	(4)
3	4	-7.250	-34.750	-85.500	10073.000	0.419	515	(10)
4	4	-25.250	0.000	-85.500	9219.500	0.400	3172	(5)
5	27	17.889	-17.259	-97.333	9339.813	1.399	119	(6)
6	9	-24.111	-31.333	-83.000	9042.333	0.405	387	(3)
7	4	8.500	-28.750	-82.250	9808.250	7.090	1076	(8)
8	5	-23.000	3.000	-85.800	9278.400	0.400	636	(5)
9	3	2.667	-20.000	-72.000	8634.000	0.419	2077	(1)
11	3	12.000	-29.000	-82.333	9782.333	6.909	471	(8)
12	3	-18.333	-26.667	-97.667	10069.000	1.690	1276	(9)*
16	3	-9.667	-32.000	-85.667	10072.000	0.419	377	(10)
19	3	-2.000	-20.333	-72.000	8647.000	0.418	830	(1)
20	6	19.667	-18.333	-96.833	9339.999	1.388	370	(6)
21	10	31.600	-33.900	-98.100	9426.400	1.403	532	(7)
22	6	-6.833	-35.000	-85.667	10068.500	0.419	749	(10)
23	4	10.750	-25.500	-82.000	9761.250	7.010	403	(8)
24	5	0.000	-21.200	-72.000	8708.600	0.419	1247	(1)
26	3	-23.000	-31.333	-83.667	9057.000	0.402	386	(3)
28	7	18.286	-18.429	-97.143	9341.143	1.417	563	(6)
30	4	11.000	-27.250	-82.500	9794.750	6.995	810	(8)

Design and Performance of a Programmable Analog Neural Computer

Paul Mueller, Jan Van Der Spiegel, David Blackman,
Peter Chance, Christopher Donham, Ralph Etienne-Cummings

Corticon Inc.
3624 Market Str., Suite 508
Philadelphia, PA 19104

Abstract: A prototype programmable analog neural computer has been developed and assembled from over 100 custom VLSI Modules containing neurons, synapses, routing switches and programmable synaptic time constants. The modules are directly interconnected and arbitrary network configurations can be programmed. Connection architecture as well as neuron and synapse parameters are controlled by a digital host. Connection symmetry and modular construction allow expansion of the network to any size. The network runs in analog mode. Network performance is monitored by the host through an A/D interface and used in the implementation of learning algorithms. The machine is intended for real time, real world computations. In its current configuration maximal speed is equivalent to that of a digital machine capable of 10" FLOPS.

Through simple feedback connections in conjunction with synaptic time constants the neurons can be transformed into spiking units resembling biological neurons and networks of such units can be used in simulations of small biological neural assemblies.

Programming software has been developed and several small applications have been implemented.

Because of the programmable synaptic time constants the machine is especially suited for real time computation of temporal patterns as they occur in speech and other acoustic signals. Networks for the dynamic decomposition and recognition of acoustical patterns including speech signals (phonemes) are described. The decomposition network is loosely based on the primary auditory system of higher vertebrates. It extracts and represents by the activity in different neuron arrays the following pattern primitives: Frequency, Bandwidth, Amplitude, Amplitude Modulation, Amplitude Modulation Frequency, Frequency Modulation, Frequency Modulation Frequency, Duration, Sequence.

The frequency tuned units are the first stage and form the input space for subsequent stages that extract the other primitives, e.g. bandwidth, amplitude modulation etc. for different frequency bands. Acoustic input generates highly specific, relatively sparse distributed activity in this feature space, which is decoded and recognized by units trained to specific input patterns such as phonemes or diphones.

A larger machine, with much higher component count, speed and density as well as higher

resolution of synaptic weights and time constants is currently under development. Some specific design issues for the construction of larger machines including selection of optimal component parameters, high density interconnect methods and control software are discussed.

1.0 Introduction

Most current work on Neural Networks and Neural Computation involves software simulations using digital computers. Simulations of larger problems can be very time consuming, especially if the network includes recurrent connections or deals with dynamic problems as they occur in the computation of acoustical signals or motion. Synaptic connections in such networks have both a weight and a transfer time constant and their operations are described by large numbers of simultaneous non-linear differential equations. Numerical evaluations of such systems on digital machines is inevitably slow requiring iterative solutions with multiple time steps for each connection. Biology avoids this problem by analog computation which allows simultaneous and continuous processing of multiple time varying inputs to each neuron. Neural networks of this type can be implemented in analog VLSI but so far have been restricted to small and mostly task specific systems [1-6]. As part of an effort to build a larger programmable system, we have assembled and tested a prototype Analog Neural Computer in which the network architecture as well as neuron and synapse parameters are programmable [7,9]. The machine is designed for real-time computation of neural network problems. It was intended to validate the design concept; a larger machine for practical applications is currently under development.

2.0 System Overview

The computer contains directly interconnected arrays of electronic neurons, modifiable synapses, modifiable synaptic time constants and analog routing switches. The arrays are fabricated on VLSI chips that form separate modules and are mounted on interconnected circuit boards. Neuron arrays are arranged in rows and columns and surrounded by synapse and routing switch arrays. The switches select the connections between neurons. Fig. 1 illustrates the machine architecture; a photograph of the machine is seen in Fig. 2.

The computer runs in analog mode. However, interconnections, synaptic gains, time constants and neuron parameters are set by a digital host computer through an interface board either from the keyboard or from stored programs. For the implementation of learning algorithms and the display of neuron activity, selected time segments of the outputs from all neurons are multiplexed, digitized and stored in host memory. The multiplexing operation is independent of, and does not interfere with the analog computations of the network.

Adjustments of synapse parameters and connection architectures are computed by learning algorithms on the basis of the stored neuron activity and loaded serially into the neural computer via a digital interface board.

The machine has several unique attributes. First, both the gains and the time constants of synaptic transfer are separately programmable, a feature that is indispensable for analog computation of time domain phenomena such as motion, target tracking and analysis of acoustical patterns. Second, it is constructed from interchangeable modules with two-dimensional symmetrical pinout that permits easy modification of the numerical ratios and positions of neurons, synapses and routing switch modules. Third, the network is expandable to arbitrary size. Finally, the coupling of analog and digital hardware adds the flexibility of digital methods.

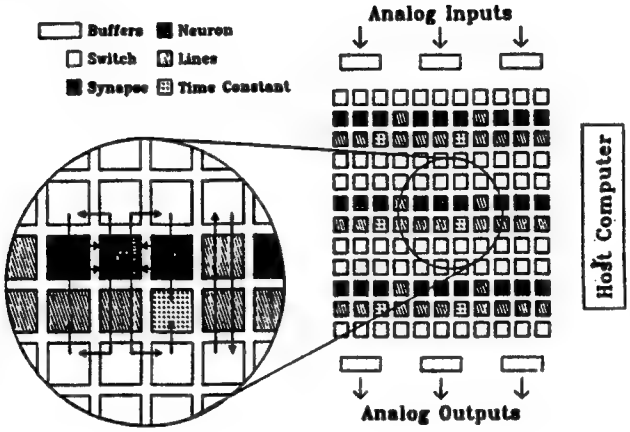


Fig. 1 General architecture and data flow. Neuron modules receive inputs from synapse modules east and west and feed outputs into switch modules north and south. Due to the connection symmetry the network can be expanded to arbitrary size.

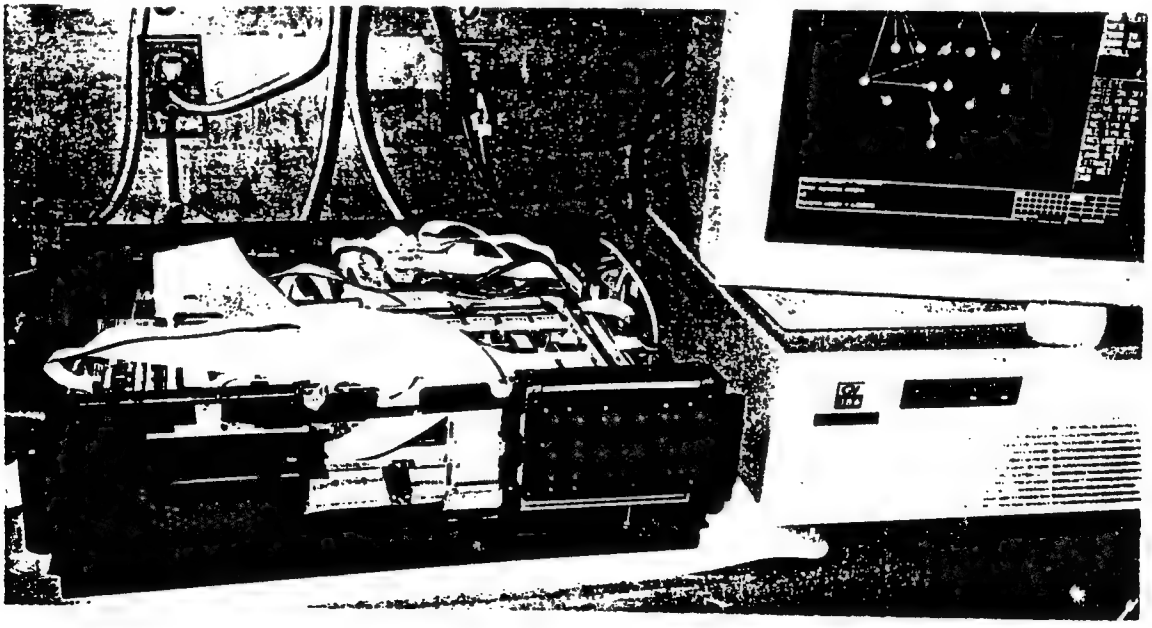


Fig. 2. Photograph of the neural computer with cover removed, showing the main circuit boards, power supplies and LED display panel. The host computer is at right.

In its current configuration the machine is composed of 3 circuit boards each containing 33 directly interconnected VLSI modules. Each module contains a separate array of neurons, modifiable synapses, modifiable synaptic time constants or analog routing switches. The neurons have an adjustable threshold and minimum output at threshold, synapse gains are programmable from -15 to +15 with logarithmic 6 bit control and the time constants from 1 ms to 2 s with 5 bit resolution.

The different VLSI modules containing arrays of neurons, synapses, time constants or routing switches were fabricated in 2u CMOS. For the prototype the synapse chip contained 8xl6 synapses, the switch chip 16xl6 analog crosspoint switches and the neuron chip 8 neurons. The relatively low component count was dictated by economic considerations. For details of the VLSI components see ref. 7. and 9.

The component count of the modules for the next generation machine have been scaled up by a factor of four, with an 8 bit synapse weight resolution and a time constant range from 1 us to 1 s with 6 bit resolution. The time constants are located on the switch chips.

3.0 Description of the Modules

3.1 The Neuron Module

The module contains 8 neurons, an analog multiplexer and digital control logic for addressing the chip and controlling the multiplexer. The neuron input is a current provided by the synapse and the output a voltage (0-4V) . The neuron transfer function is shown in Fig. 3. The reasons for choosing this particular transfer function which permits variable degrees of arithmetic and logic operation by the same unit have been described elsewhere [7,10]. In order to avoid possible crosstalk in the switch lines the neuron bandwidth was limited to 300 Khz and provisions were made to further reduce the bandwidth at the system level. However no crosstalk problems were encountered and in future designs the bandwidth shall be increased to 1 Mhz. The analog multiplexer, which permits sampling of the neuron activity by the host computer, is of conventional design and does not interfere with the neuron operation. The sampling rate of the multiplexer is under software control with a maximal clock rate of 150 Khz. Thus the activity of all neurons can be sampled at up to 0.5 ms per point.

Because of the large impedance of nerve axons, biological neurons communicate their output value via a pulse frequency code which is generated by voltage dependent ion channels in the cell membrane. It has been argued that this mode of operation has computational significance through phase locking or generation of special pulse patterns and electronic neuron analogues with spiking outputs have been implemented using discrete components or VLSI circuits [11-12]. In our system each neuron can also be programmed to operate in this mode and the entire machine can be transformed to operate as a neural simulator for biological research. This is accomplished by delayed inhibitory and undelayed excitatory feedback which mimic the action of ion channels in biological neurons. An

example is shown in Fig. 4. However, we have not yet seen any compelling need to use this mode for computational purposes.

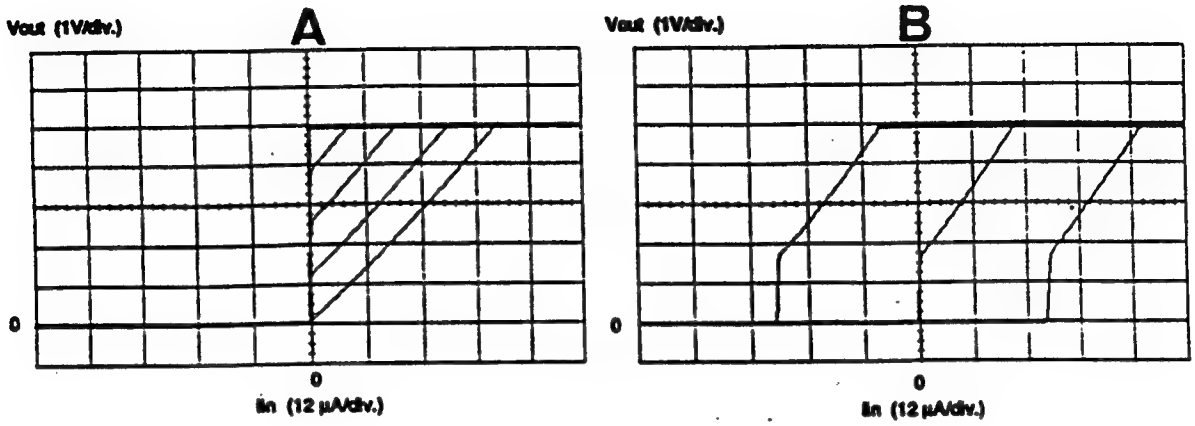


Fig. 3. The measured neuron output voltage versus input current: a) transfer characteristics for different minimum outputs at threshold ranging from 0 to 5 V. b) characteristics for different threshold bias currents: -30, 0, and +30 uA with a 1.7 V step at threshold.

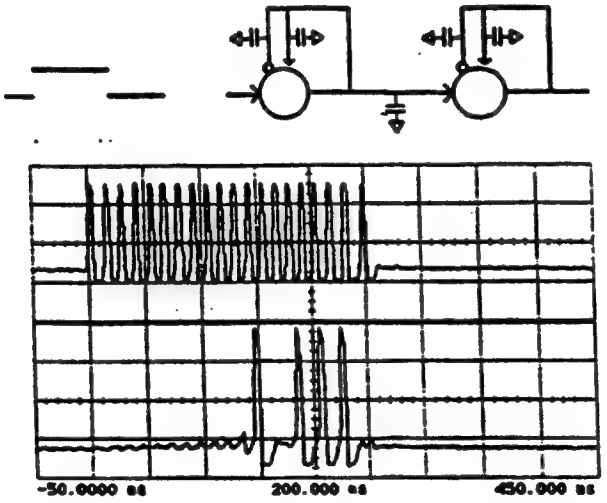


Fig 4. By employing feedback connections as shown in the top diagram, the neuron input-output function can be modified under program control to resemble spiking biological neurons. The feedback time constants determine spike duration and refractory period. As in a biological neuron the spike frequency is controlled by sum of input currents. The top record shows the response of the first neuron to a square wave input. The bottom record shows the response of the second neuron to the input from the first neuron. The synaptic time constant of the inter-neuron connection determines the rate of rise of the postsynaptic

potential and therefore the transfer delay.

3.2 The Synapse Module

The synapse converts the neuron output voltage into a current through a V/I converter and scales the current with current mirrors into 5 values. A current recombination unit controlled by a shift register memory combines the currents to the appropriate value specified by the user programmable weight. In order to achieve a large dynamic range for the weights (3 orders of magnitude) a floating point scheme was chosen for the current scaling and recombination. The block diagram of the synapse module and measurements of synapse performance are shown in Figs. 5 and 6.

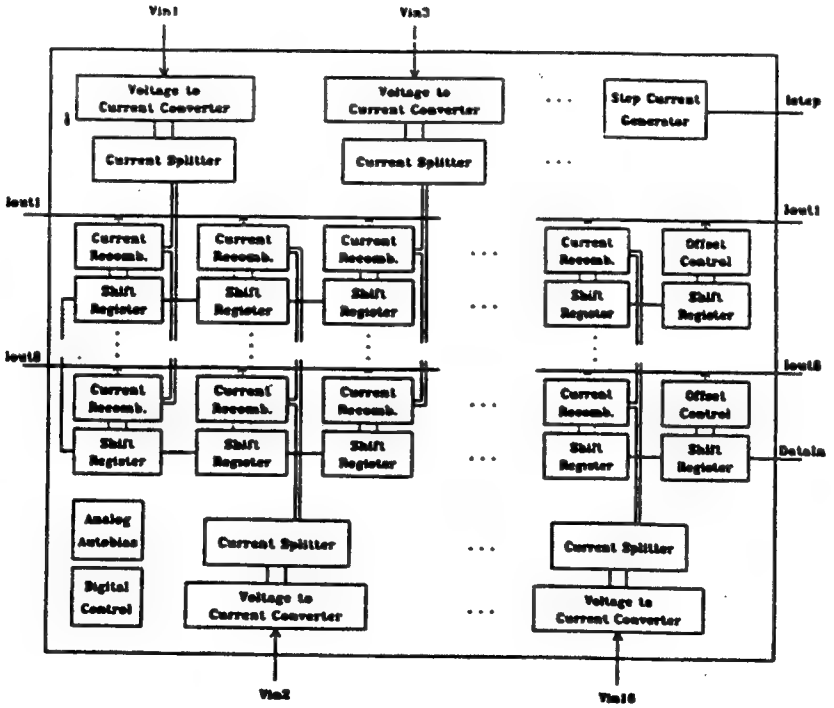


Fig 5. Block diagram of the synapse module; the inputs are common to all synapses in the same column and the output lines sum the currents of all synapses in the same row.

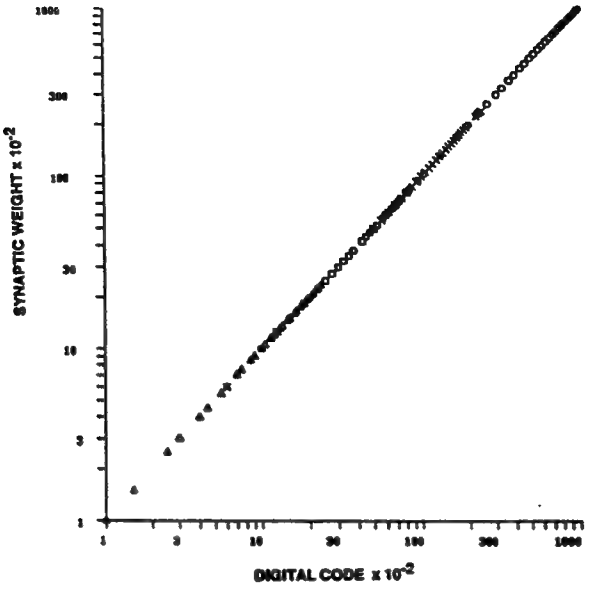


Fig 6. The weight factors of the synapse versus the digital code programmed in synapse memory.

3.3 Programmable Time Constants

For time domain operations, programmable time constants over the range from 1 us to 1 s are desirable. These time constants were implemented fully on chip with no external capacitors by using an operational transconductance amplifier as a high resistance. The time constant values are set by a 4 bit local memory. The values vary logarithmically with digital code as seen in Fig. 7. In the next generation machine the time constants are placed on the switch chips and have a 5 bit value control and 1 bit direction control.

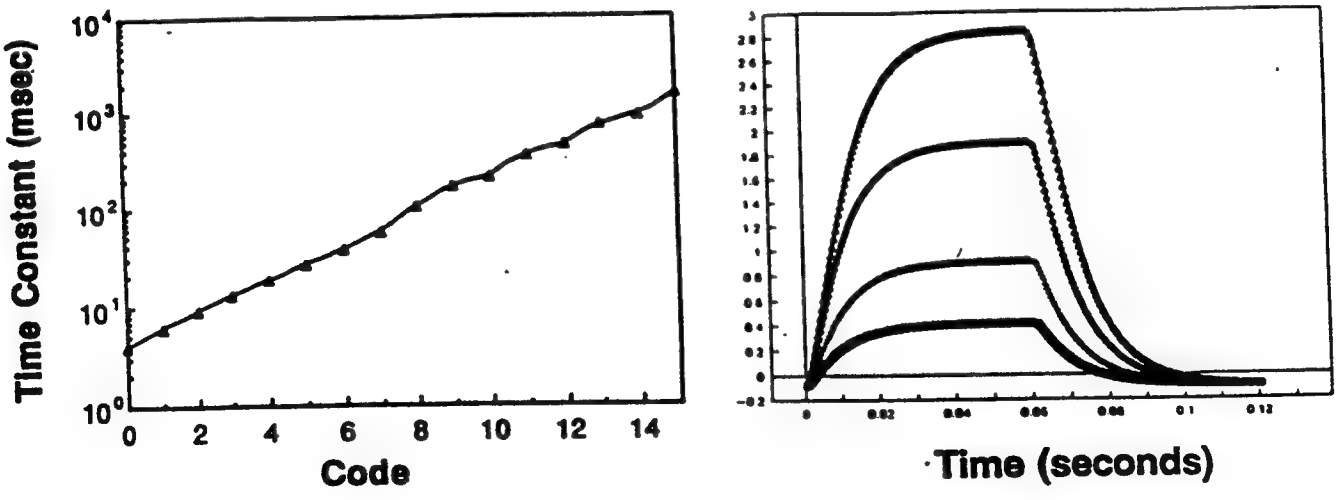


Fig. 7. The average time constant values versus the digital code measured from 16 circuits on 4 different chips. The average variation ranged from 2.5 to 7.5%. The records show the response to square waves.

3.4 The Switch Module

This module is used to route the analog signals between the neurons and the synapses and thus specifies the network architecture. The module consists of a 16 x 16 array of crosspoint switches with additional cut switches at the borders. Each crosspoint switch is set by a local one bit memory, the cut switches also allow grounding either output or input and have a two bit memory. On-resistance is 2 to 3 KOhms and Off-resistance > 1 TOhms. The block diagram of the module is shown in Fig. 8. In the next machine the array size is 32 x 32.

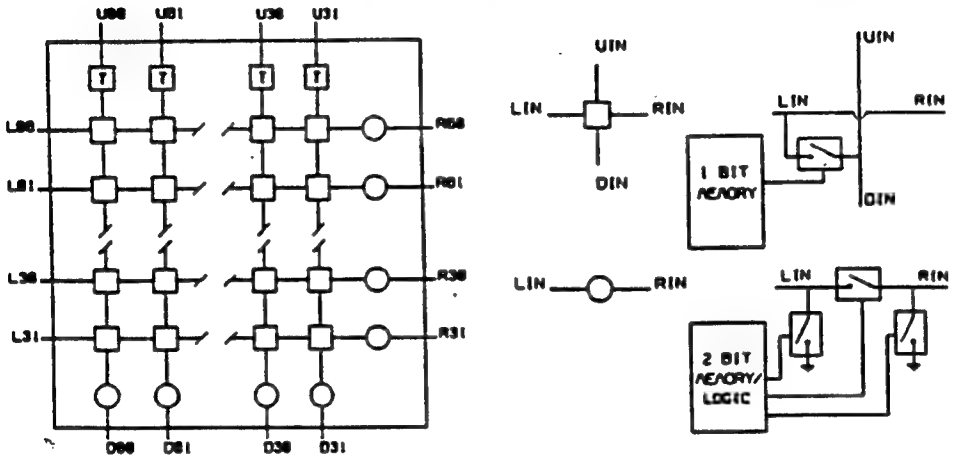


Fig 8. Block diagram of the switching fabric.

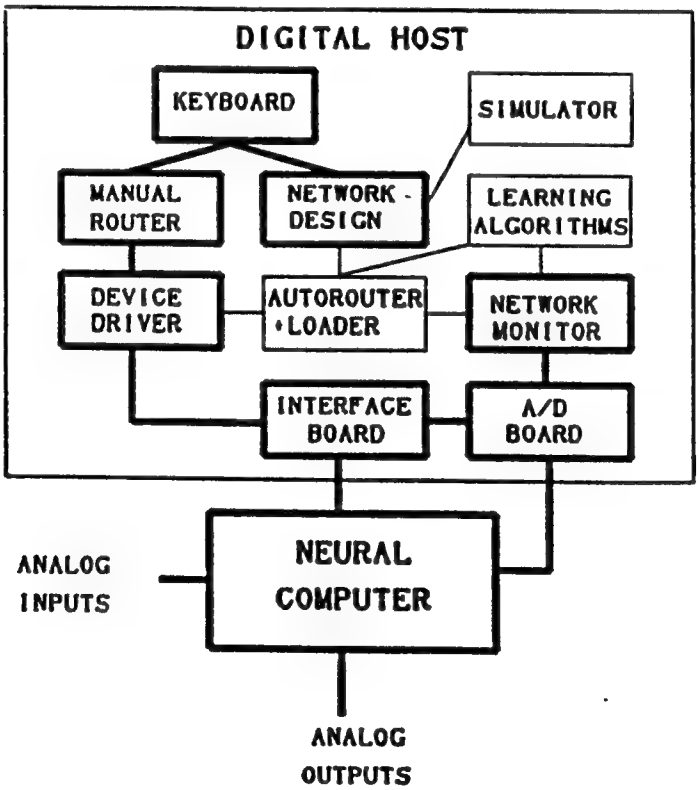


Fig 9. Overview of the software for the neural computer

4.0 System Operation

The machine is controlled by software residing in the digital host. An overview of the different tools is shown in Fig. 9. The connection architecture, synapse and neuron parameters are set and verified serially at 2 Mhz from a digital host through a special interface board and graphic control software. At the lowest level of control the network parameters are set manually through a mouse and graphic display that highlights the connections and displays the synapse parameters. Parameters are loaded either for the entire network, a selected board or a selected module.

Higher level control software, allowing the automatic compilation of conceptual networks into the machine has also been developed. In this case the neurons in the conceptual network are first partitioned and grouped according to the degree to which their inputs originate from common sources. The groups are then assigned to specific modules in the physical network by a placement routine and subsequently interconnected by an autorouting program based on min-cut procedures. This program generates settings of switches and component parameters that are loaded into the neural computer.

The network is connected directly to the outside world through parallel analog I/O buffers. There are 352 (2048 for the large machine) analog I/O lines available. In addition to the analog I/0, each neuron chip contains an analog multiplexer that enables the digital host to monitor and store the neuron activity for graphic display and the implementation of programmed learning algorithms. The multiplexer feeds a 250 Khz A/D board located in the host over a common line. The monitor software generates either separate graphs of selected time segments of each neuron output as shown in Fig. 11, or it generates a continuous gray scale display of activity for all neurons. Neuron activity is also displayed directly by an LED Panel seen in Fig. 2.

The neuron activity patterns read by the host can form the basis for implementation of different learning algorithms in which the network and the host form a closed loop. The learning of a simple OCR task by backpropagation has been demonstrated

5.0 Performance of Simple Computational Tasks

In order to evaluate its performance, the machine programmed for several tasks. A few examples which "Winner Take All" net, an associative net and a network for optical character recognition have been discussed elsewhere 14. In all cases the actual neural computations (not the learning) are performed in real time, with response time limited only by the 300 Khz bandwidth of the neurons. Signal to noise ratio for all operations is better than 10 bits.

6.0 Decomposition and Recognition of Acoustical Patterns

The machine is especially suited for the real-time analysis of dynamic patterns such as

speech or other. Acoustical patterns. Many aspects of such patterns involve time as an explicit variable and their decomposition demands large computational power.

The early stages of higher vertebrate auditory systems transform acoustical signals into patterns of neural activity distributed in space and time and decompose the relations between these variables into their pattern primitives. Examples of such primitives are frequency, amplitude, frequency and amplitude modulation, amplitude modulation frequency or rate, signal duration and sequence. These primitives are represented separately and form the basis for the recognition of more complex signals [13, 15-16].

We have programmed a network that performs some of these decompositions, albeit at low resolution of the variables. The outputs of the decomposition neurons are then decoded by specially tuned neurons that respond to different phonemes within a word and in conjunction with the host computer provide a real-time phonetic printout of speech. The architecture of the decomposition network is seen in Fig.10.

At the input to the net, sound is decomposed into its different frequencies by a set of 8 bandpass filters (200 - 3000 Hz). The rectified filter output is fed into the first stage of neurons which extracts local maxima of amplitude (E) vs space (S), (i.e. frequency) through antagonistic center-surround organization of the inputs combined with lateral inhibition. This operation computes the amplitude contrast (d_2E/dS_2) across the frequency spectrum and normalizes the output of the first stage neurons with respect to the sound amplitude.

The outputs of the first stage neurons feed into the next stage which computes the changes of energy (E) with respect to time (dE/dT) at each frequency. Positive and negative values of dE/dT are represented by outputs from separate neurons ("ON" and "OFF" units). This computation is achieved by a combination of direct and delayed (low pass filtered) inputs of opposite polarity from frequency tuned units to the "ON" or "OFF" units. For the "ON" units the excitatory connection is direct whereas the inhibitory input is low pass filtered. The "OFF" units receive inputs of opposite polarity.

The "ON" and "OFF" units represent not only the temporal boundaries of events, i.e., beginning and end of a particular sound of a certain frequency, but also provide a measure of time from the temporal boundary by transforming time into an exponentially decaying potential. In this sense they act as an analog clock and provide the basis for the computation of formant transitions (motion), sequence and duration. "ON" and "OFF" units are commonly found at the early stages of biological sensory systems.

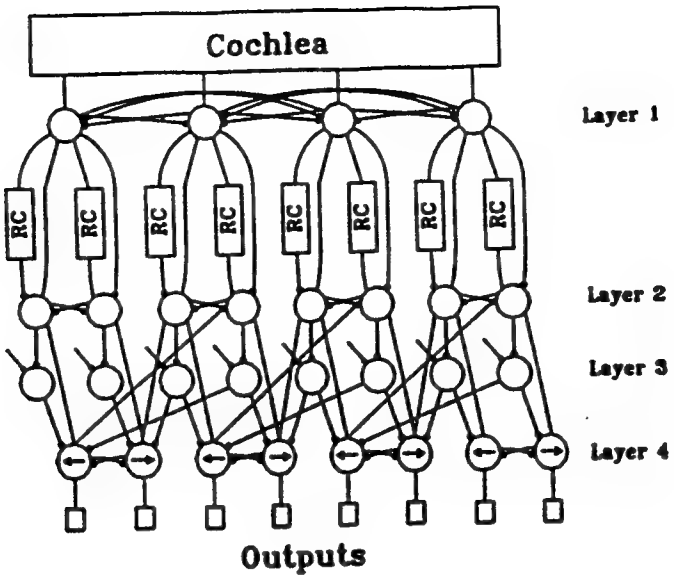


Fig. 10 Part of the conceptual design of a network for the primary decomposition of acoustical patterns. The primary neurons receive rectified inputs from eight band pass filters (Cochlea) in an antagonistic center-surround scheme. These neurons are interconnected with mutually inhibitory inputs with spatially decaying gains. They extract the maxima of the sound amplitude. The next stage extracts separately the temporal rise and decay of the sound amplitudes. These neurons receive delayed excitatory or inhibitory inputs from the previous stage. The third stage neurons are normally "on" through a positive bias input and compute the complement of the activity of the second stage neurons. The fourth stage units compute the changes of frequency maxima and their direction (arrows) through a combination of the second and third stage neurons. In essence they are motion detectors.

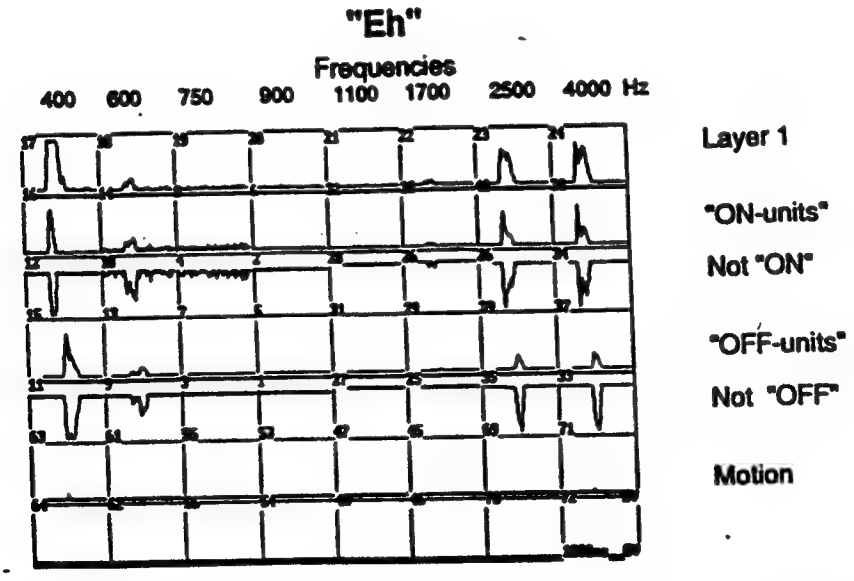


Fig. 11 Neuron outputs from the network shown in Fig. 10 for the phoneme "Eh". Each block shows the output of one neuron as function of time during the pronunciation of the phoneme. The columns represent increasing frequency from left to right (from 400 to 4000

Hz). The activity of the neurons in the first row (Layer 1) is proportional to the contrast function of the sound energy at the different frequencies. The second and fourth rows represent the positive and negative rate of change of the neuron outputs in layer 1 at the different frequencies and the third and fifths rows are the complement of rows two and four. The outputs of neurons in rows five and six (labeled "Motion") represent up and down changes of frequency as function of time as they occur, for example, in diphones during formant transitions. There is no activity for this phoneme.

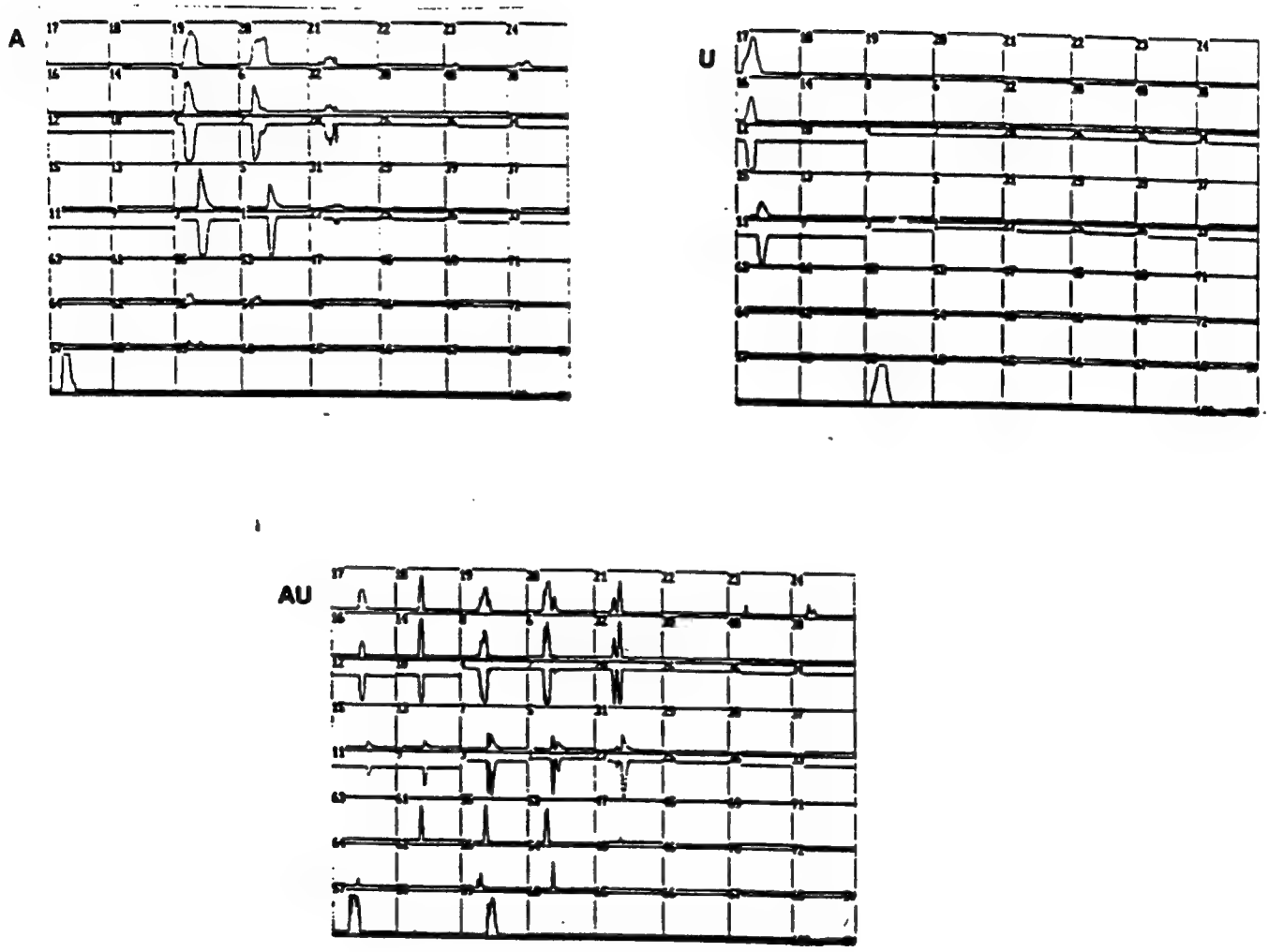


Fig. 12. Neuron outputs in response to the isolated phonemes "A", "U", and the diphone "AU". The decomposition network was identical to the one used in Fig. 11. In addition other neurons were added that were programmed to decode the activity patterns in the decomposition network in response to a particular phoneme. Neuron #57 decodes for "A" and neuron 59 for "U". During pronunciation of "AU" these neurons become sequentially active. In addition neurons 61, 55 and 53 indicate the shift of the formant frequencies. These neurons are used for the decoding of this diphone. Activities of the decoding neurons

are used by the digital host to print in real time the spoken phonemes.

The second temporal primitive extracted by the net is the change of frequency that occurs for example during formant transitions associated with certain diphones. Since each frequency band is represented by the activity of a separate neuron group, changes of frequency appear as "motion" of activity in neuron space, i.e., +/-Ds/Dt. We define this motion as the decay of activity at one position and increase of activity at a neighboring position and compute the rate and direction of motion (increase or decrease of frequency) by neural "AND" gating of the outputs from neighboring "ON" and "OFF" units. This "AND" gating is achieved by implementing a "NOT(NOT)" function of the "ON" and "OFF" units through separate representation of, and inhibition by their complementary output 10 The motion-tuned neurons are sensitive to direction, amplitude and rate of motion. The range of rate sensitivity is determined by the time constants of the "ON" and "OFF" units which in this case were adjusted to the natural rates of frequency change during speech formant transitions.

Some examples of the network performance are shown in Figs. 11 - 13. Due to limitations of the number of available neurons, only a few neurons were tuned to the activity patterns of selected vowels, diphones and consonants. These neurons responded in real time during the pronunciation of the phonemes even if spoken in context with other phonemes. Diphones such as "AU" or "I" were recognized as entities and not as their separate components. Although statistics were not taken, recognition was to a large extent speaker independent. The phoneme decoding and recognition used essentially convolution-based strategies that have been reported elsewhere [10]. Gradient descent learning algorithms that take the temporal characteristics of the patterns into account are under development.

7.0 Discussion

Because of the limited size of the prototype machine, the pattern decomposition was restricted both in spectral resolution and in the number of primitives that were represented. In a larger machine it would be desirable to increase the spectral resolution to at least 32 frequencies and to represent different amplitude ranges by different neural arrays. In addition a limited range of signal durations, sequences and perhaps amplitude modulation frequencies should be extracted for the primary frequencies. Circuits that perform these operations are relatively simple and have been described elsewhere.

The circuits and their parameters shown above were designed "by hand" and required only minor trimming for optimal performance. Appropriate learning algorithms that deal with such continuous-time temporal pattern processing nets have yet to be developed. The machine architecture permits in-loop or out-of-loop learning using the digital host for weight adjustment on the basis of stored algorithms and typical backpropagation methods have been employed for the learning of simple OCR tasks. For learning in feedforward nets the system offers no speed advantage over digital simulations. However in feedback nets the response to training inputs patterns occurs in real time and this fact adds significantly to the learning speed.

8.0 Commercial Applications

As discussed above, even the fastest currently available digital machines are inadequate for coping with the staggering computational demands imposed by the massively parallel nature of neural algorithms. Machines modeled after biological brains will have computational power surpassing by many orders of magnitude the capabilities of digital machines. Initially their availability would enable engineers and scientists to realize the potential provided by neural algorithms. For example, the machine could be used as a simulation and development tool for the design of dedicated neural VLSI systems such as smart sensor arrays for vision or speech. Such systems would perform extensive processing such as acoustical pattern decomposition as discussed above or character recognition, visual motion tracking or other functions that need little or no programmability. In this application the Neural Computer is used to design and test particular neural circuits that are subsequently implemented in silicon from standard components - Neurons, Synapses, Timeconstants - that are stored either as circuits or layouts in a library.

But the commercial potential of these machines is not limited to their immediate role as research tools. Eventually they could find widespread use in manufacturing, communication, service and defense industries. In the Government segment the product would be applicable in the defense sector in areas already mentioned, such as target acquisition, autonomous vehicle control or logistics and may eventually reduce the reliance on manpower.

In the private sector the machines would find applications in speech recognition, machine vision, robotics and manufacturing control.

At this stage it is difficult to make an accurate estimate of the near term commercial potential. Much depends on the rate at which this technology develops and becomes accepted. In the long term, the world market for this type of product could be very large. Based on current activities in this field and experiences at conferences and trade shows, the product is expected to find a receptive and growing market.

9.0 Acknowledgments

This work was supported by grants from the Office of Naval Research and the National Science Foundation.

10.0 References

1. Mead, C., Analog VLSI and Neural Systems, Addison-Wesley, Reading, MA 1989.

2. Alspector,R., Allen, B., A Neuromorphic VLSI Learning System, Advanced Research in VLSI, Proc. of the 1987 Stanford Conf.

3. Liu, W., Andreou, A., Goldstein, M., An Analog Integrated Speech Front End based

on the Auditory Periphery. Proceedings of the 1991 IJCNN II 861 - 864.

4. Holler, M., Tam,S., Castro,H., Benson,R., An electrically trainable artificial neural network with 10240 floating gate synapses. IJCNN V 2 , 191 - 196 1989.

5. Boser, B., Sackinger,E., An analog neural network processor with programmable network topology, IEEE ISSCC Digest of tech. Papers, 184*-185 1991.

6. Graf,H., Henderson,D. A reconfigurable CMOS Neural Network, IEEE ISSCC Digest of Technical Papers, 144*-145 1990.

7. Mueller, P., Van Der Spiegel J. et al. Design and Fabrication of VLSI components for a General Purpose Analog Neural Computer. IEEE Workshop on VLSI and Neural Systems, Portland 1989. C. Mead Ed. Kluwer, Boston Ma.

8. Mueller, P. Van Der Spiegel, J. et al. Design and Performance of a Prototype General Purpose Analog Neural Computer, Proceedings of the IJCNN Seattle, WA, 1991.I 463 -468.

9. Van Der Spiegel, J., Mueller, P., Blackman, D., Chance, P., Donham., Etienne-Cummings, R., Kinget., P., An Analog Neural Computer with Modular Architecture for Real-Time Dynamic Computations. IEEE Journal of Solid State Circuits, 27, 82 -92. 1992.

10. Mueller, P., Lazzaro, J.A machine for neural computation of acoustical patterns with application to real time speech recognition, AIP Conference Proceedings, 151:321-326, 1986.

11. Mueller, P., Martin, T., Putzrath, F., General Principles of Computation in Neural Networks with application to acoustical pattern analysis, Biological Prototypes and Synthetic Systems, New York: Plenum, 1962, 192-212.

12. Mahowald, M., Douglas, R., A Silicon Neuron, Nature, 354; 515 - 517, 1991.

13. Mueller, P., Computation of Pattern Primitives in a Neural Network for Acoustical Pattern Recognition. Proceedings of the IJCNN 1990 Wash. I 149-151.

14. Mueller, P. Van Der Spiegel, J., Agami, V., Blackman, D., Chance, P., Donham, C., Etienne-Cummings, R., Flinn, J., Massa, M., Samarasekera, S., Design and Performance of a Prototype Analog Neural Computer. Proc. of the 2nd International Conference on microelectronics for Neural Networks, Kyrill and Method Verlag Munich 1991, 347-357.

15. Evans, E.F., Neural processes for the detection of acoustic patterns and for sound localization. In: The Neurosciences 3rd Study Program, F.O. Schmitt , F. Worden Eds., MIT Press Cambridge Ma. 1974, 131-145.

16. Suga, N., Auditory Neuroethology and speech processing. In: Auditory Function, Neurobiological Bases of Hearing. G. Edelman, E. Gall, M.Cowan, Ed. New York: Wiley, 1988, 679 -720.

Application Of An Adaptive Clustering Neural Net to Flight Control of a Fighter Aircraft

James C. Smith*
Robert V. Walters**
Henry R. Jex*
Bimal L. Aponso*

Abstract: An Artificial Neural Network (ANN) controller was developed for a typical fighter aircraft's longitudinal Stability and Control Augmentation System (SCAS), using elevator and thrust-vector-angle controls, and operating at high angle of attack. The "baseline" neurocontroller (NC) was in the feedforward loop, with inputs from the pilot's pitch rate commands, and from SCAS feedback error. An Adaptive Clustering Network (ACN) algorithm (developed by STR Corporation) was used to train radial-basis-function neurons to imitate the inverse of plant dynamics. Significant improvements in performance resulted from the NC action, and these effects were analyzed and interpreted by frequency-domain describing functions. Thrust vector failures were handled satisfactorily, but reconfiguration of the SCAS was not possible within the simplified aircraft and NC simulation. Emphasis of the analysis was on validation methods and ways to interpret the time-varying non-linear NC effects. Recommendations for future research include: ways to choose sensed signals so that neural net can more efficiently identify the failures of correlated control effectors; the further use of frequency-domain describing functions to identify neurocontroller dynamic processes and stability; and the evolution of a Neurocontroller Analysis Toolbox containing: diagnostic forcing functions, test methods, analyses, and benchmark criteria for evaluation to a common NC standard.

This work was sponsored by a Phase I SBIR award through the Naval Air Development Center, Warminster PA Contract number N00019-90-RCB7A4R. Contract Technical Monitors were Robert D. Digirolamo and Marc L. Steinberg.

1.0 Introduction

This project began as an effort to demonstrate the utility of Artificial Neural Network (ANN) technology in adaptive flight control design, specifically for advanced reconfigurable control systems in which control strategy is, in part, determined by mission requirements and variations in pilot maneuvering styles. It was originally proposed that ANNs would be of considerable value in allowing the control system to adaptively organize itself during flight with emphasis on its ability to detect failures and reconfigure the control system. Termed an Intelligent

*Systems Technology, Inc., Hawthorne, CA

**STR Corporation, Reston, VA

Configuration Management System (ICMS) it would be fast enough to catch a serious control failure in the middle of a maneuver to prevent loss of the aircraft. The architecture of this system was based on a forward modeling application reported by Jordan (Ref. 7), and embodied a multi-layer perception (MLP) which adapted by means of backpropagated errors. Comprehensive treatment of this problem proved to be beyond the scope of Phase I funding. Additionally, the oscillatory behavior of the MLP during learning was thought to be inappropriate for this aircraft scenario. The Phase I objectives were revised accordingly (Ref. 1). The scenario was still the high angle-of-attack maneuvering of a fighter in the pitch axis, wherein *both* an all-moving stabilizer *and* a set of thrust-vectoring paddles would be required to achieve the tracking bandwidth needed for satisfactory flying qualities, especially at low-speed, high-angle-of-attach flight conditions.

An analysis was made on the challenging problem of detecting which of the two aft-located controls had failed. This analysis revealed that a novel combination of bandpassed acceleration and pitch-rate signals, related to the instantaneous center of rotation of each control, would be best. Specific conclusions are given in Ref. 6 contained in Appendix A of (Ref. 1). It was felt that the required aircraft simulation was more complex than was practical for Phase I, so the demonstration of the detection/reconfiguration feature was deferred to Phase II.

2.0 Revised Phase I Neurocontroller Concept

STR and STI considered several control architectures as candidates for this research. STR also experimented with several alternative types of neural elements and made observations of speed adaptation and open loop behavior. Three of these architectures are shown in Fig. 1. Figure 1-a illustrates the principle of control using "Direct Inverse" mapping. The objective, here, is to make $Y_N \equiv 1 / Y_C$ so that the controller/vehicle transmissibility is $Y_N \cdot Y_C = 1.0$, i.e., perfect following. Although this approach can yield a controller with attractive properties, it is rarely successful in application because plant dynamics often do not have a realizable inverse, or the plant inverse $Y_N = 1/Y_C$ cannot be quickly identified.

Figure 1-b illustrates a "Forward Model Controller" based on Ref. 7 (Jordan). We considered this to be a promising architecture for our approach. After further analysis, it was revealed that the adaptive element exhibited a general property of slow adaption due to the use of backpropagation of errors, and, in some cases, had failed to converge on a solution so it was not used.

Figure 1-c shows the "feedback error controller" described by Kawato (Ref. 2). The concept of using a combined feed forward/feedback controller had initial appeal owing to the prospect of rapid initial adaptation without stressing reasonable envelope constraints.

Normally, estimating $Y_C(s)$ using a backpropagation type of ANN would be too time consuming. Kawato, *et al* (Ref. 3), following some principles related to the human control of limbs, while developing ANN controls for robot arms, had made the following qualitative observation that was analytically verified in Ref. 1:

i. As can be shown by the fundamental laws of open- and closed-loop systems, when the feedback error (command output) is made much smaller than the command, by a "tight" feedback control law, then the output closely resembles the command.

ii. Consequently the control laws for such a tightly closed-loop will yield control signals that are close to those needed to invert the controlled elements' response over the closed-loop bandwidth; and result in a stable system.

iii. Therefore, the control commands from the feedback error controller approximate $1/Y_C$.

So, the on-line neural-net controller can be trained to map these nearly "ideal" control signals that resulted in the small control errors. This further reduces the errors, so, ideally, the NC would converge to the case $Y_N \cong 1/Y_C$ with minimal error on the closed loop bandwidth. Finally, rapid forward-propagation-learning could be used!

Since the fighter case we selected already had a sophisticated stability and control augmentation system (SCAS) present, the control commands from the error feedback were already available, so we settled on this type of Kawato neurocontroller for our Phase I demonstration example (see Fig. 2). A number of modified objectives were laid down, in consultation with our Contract Technical Monitor, and they are given in the Section I of Ref. 1.

The type of network architecture applied to this loop structure turned out to be a major side issue, since traditional network topology (based on use of backpropagation) would not adapt rapidly enough to be suitable for the needs of real-time flight control. For this application, STR selected a proprietary network design which they had previously used on pattern recognition problems. The design of this type of network, referred to as an Adaptive Clustering Network (ACN) bears a functional resemblance to networks based on Adaptive Resonance Theory (ART) implemented with supervised learning like ART, ACNs offer a compromise solution to the problem of maintaining adaptive potential in a near stable environment. The ACN model is a three-layer structure which sandwiches a hidden layer of adaptable size between a simple fan-out input layer and a Widrow-Hoff output layer in which weights are adjusted to minimize a computed error function.

ACN operation allows it to add (or delete) hidden layer elements as required for input state-space coverage, and to cluster their receptive fields based on utilization. This yields a network whose size is adaptively governed, based on problem complexity, and which can be trained to criterion up to three orders of magnitude faster than networks based on Multi-layer Perception using Back Propagation (MLP/BP).

STR Corp implemented the aircraft-SCAS- neurocontroller within a proprietary software shell using Symantec's THINK Pascal running on a MacIntosh II CX computer. Because of their experience with dynamic ACN systems and feedback control modeling, an unusually complex piece of software was created within the budget and time constraints of Phase I funding. The STR ACN scheme was mechanized as shown in Fig. 3, with inputs from commanded pitch rate

and (derived) acceleration, and it was trained from the feedback-error-control signals.

Validating the aircraft simulation required the use of STI's FREquency Domain Analysis (FREDA) program coupled with a special form of quasi-random forcing function for efficient measurement of $Y_C(j\omega)$. This was a special set of "sum-of-sines", in which non-simple-harmonic, but integer over short runs (21 sec) sinusoid provide gentle startup/shut-down waveforms and extremely high signal/noise ratios. Analysis of this type of input yields true line spectra (so that nonlinear effects can be revealed as harmonics(s), and they have reasonably Gaussian statistics. A key feature found useful for NC training, is that the desired input state space is quickly and broadly covered in a manner similar to a continuous ensemble of in-flight tracking maneuvers.

The refinement of these "5 sum-of-sines" and "7 sum-of-sines" forcing functions will be completed in Phase II, to become a generally available and efficient tool for efficient training and testing a variety of NC system dynamic measurements. It will be shown, as the Qcmd signal in Fig. 4.

3.0 Typical Results

With these NC concepts and training techniques developed, the neurocontroller modeling efforts proceeded well, and we were able to demonstrate significant improvements in the tracking error and bandwidth with the baseline configuration. Figure 4 shows the pitch rate signals while Fig. 5 shows the control signals for NC "OFF" versus "ON" during a 42 sec training run.

Points to note in Figs, 4 and 5 are:

i. The reasonably quasirandom waveforms of the command sum-of-sines signal, which repeats every 21 sec.

ii. The running statistics settle out within a few seconds, as shown for the RMS Qerr performance measures.

ii. Comparing the Fig. 4a with 4b, turning the NC "ON" reduces the Qerr signal and the RMS Qerr by about 50 percent, as the NN mapping proceeds beyond about 5 seconds.

iv. The waveform of the Qerr signal shows that the ANN closure reduces the lower frequency components but does not affect the high frequency components.

v. From comparing Figs. 5a, and 5b the various control signal components with NC = "ON" show *larger* STABsum and VECsum control commands for the smaller errors, indicating higher effective controller gain, as will be verified later.

vi. Especially for the STABsum signal, there are higher frequency components (e.g., sharp peaks) with the NC = "ON", as one or two neurons are encountered, especially at signal extremes.

4.0 Neural Controller Utilization

Most of the Phase I report covers the validation of proper NC action and is devoted to *understanding* the dynamic operation of the NC from a controls engineering standpoint. In the process, several measures of neural net activity were evolved, and the best three are shown in the Fig. 6. These are: Fig. 6a) cumulative number of cells added by the NC as it learns; Fig. 6-b) number of cells participating in each control decision (an average of 2-4 was desired for this application) and the locus of the ACN algorithm's clustering of neurons in the NC input's phase-plane. These centroids, and or spheres of influence (their effective radii in rms normalizes the coordinates) are superimposed on a trajectory of the inputs in the phase-plane, so adequate coverage can be verified. Examples of the neurocontroller action are given in Fig. 6 for the baseline case, while learning throughout the 42 second run. It can be seen that at least one, and usually 3 or 4 neurons, are involved in every point within $\alpha \pm \sigma$ region of the centroid, while the whole input phase space was covered by only 20-30 neurons.

5.0 Frequency Domain Interpretation

We have made measurements of the neurocontroller's closed loop dynamics via a frequency domain analysis of this nonlinear, time-varying element.*** Using this technique we were able to demonstrate that the asymptotic feedforward controller $Y_N(j\omega)$ is closer to the error/input transfer function, Y_{IE} than to $1/Y_c$ as intended. Figure 7 shows this important result. A concurrent theoretical analysis provided the two "model" lines, so the action of the NC can be clearly understood (see Ref. 1).

Another interesting effect on closed-loop stability has been revealed and analyzed. Because the feedforward NC block is outside the closed-loop SCAS system, it should not affect closed-loop stability. By computationally opening the closed-loop (as done in Ref. 1 and shown as Fig. 8, it was shown that the NC loop acts as if it doubles the loop gain, thereby tightening the control loop for better performance, but drastically reducing the closed-loop stability margins. (In effect, the ANN immediately maps the feedback error's control command to the Y_N block, thereby doubling the control signal for a given error). This use of frequency domain techniques to understand a neurocontroller's effective dynamics and stability is felt to be somewhat novel for ANN analysis. It deserves further development as a tool for validating NC architecture and performance.

6.0 Thrust Vector Failures

Thrust vector control failures were tested with the baseline controller and with the modified NC. Figure 9 shows the NC effects on the pitch rates from VEC failure at 21^+ seconds while Fig. 10

***Describing functions between selected signals are computed only at the precise input line spectral components, yielding accurate data over short (21 sec) periods.

shows the corresponding control signals . Although the NC did reduce the effect of the failure, it had no means to detect the specific control which failed, so it merely increased the loop gain, as before. Reference 6 analyzes the requirements for identification of which pitch control has failed. It requires both angular and linear accelerations from properly located sensors, specific frequencies for proper and rapid identification. When a more complete 3 DOF simulation of the aircraft is available in Phase II, the proper bandpassed pitch and normal acceleration signals can be added to the NC to allow it to detect a failure and to reconfigure the control signals. Even more complex approaches have been found necessary for lateral-directional NC under battle-damage cases.

We were seduced into adding more states, willy-nilly, to the neurocontroller input, with little prior analysis of the real requirements (i.e., prior to Ref. 6). The easily available output states Q and $\dot{Q}$ were added in various conditions, but VEC failure performance was no better and sometimes worse than the baseline results (Ref. 1).

Not too surprisingly, the general conclusion from these runs is that *states which are diagnostic/corrective for inner loop failures must be included in any diagnostic NC.* The complexity of the ACN mapping and the nonlinear dynamics within it implies that a more careful pre-run dynamic "systems survey" type of analysis be given to *each major NC architectural change*. Then, only carefully selected changes should be added gradually, in order to permit understanding at each step. It is all too easy, as we have found out, to mix ever more ingredients into the NC pot, stir wildly and then try to understand what had been cooked up. The more rational approach, which we usually preach, will be our discipline in Phase II!

7.0 Conclusion

The success of a Kawato feedback error configuration and an efficient Adaptive Clustering Network ANN paradigm, in combination with a quasi random input to cover the operating conditions, led to very rapid learning. Such a scheme does, however, affect the loop stability, contrary to *a priori* intuition.

The overall conclusion from the failure mode experiment was as follows: *because this baseline architecture maps the feedback-error-based corrective controls to the neurocontroller transfer functions, it cannot easily learn that one or another control surface has failed, because there are no diagnostic-value states input to the NC.* Thus, it continues to put out corrective control commands to both STAB* and VEC* which were learned from the correctly operating feedback system while following the already-mapped command-input states. The more complex case will be solved in Phase II, using the guidelines in Ref. 6.

As Phase I came to a close, we realized that one of the most valuable results from Phase I was the combined use of some classical and some novel analysis tools for the validation and understanding of neurocontrollers. Consequently, we will redirect some emphasis in Phase II to the further development of a "Neurocontroller Analysis Toolbox" for common use which would contain the above recommendation with diagnostic forcing functions, methods, analyses, and

benchmark criteria for evaluation to a common NC standard.

8.0 References

1. Smith, James C., Henry R. Jex, Robert V. Walters, and Bimal L. Aponso, Application of an Adaptive Clustering Network to Flight Control of a Fighter Aircraft, Systems Technology, Inc., TR-1285-1, Nov. 1991.

2. Kawato, M., "Computational Schemes and Neural Network Models for Formation and Control of Multijoint Arm Trajectory," Neural Networks for Control, M.I.T. Press, Cambridge, MA, 1990.

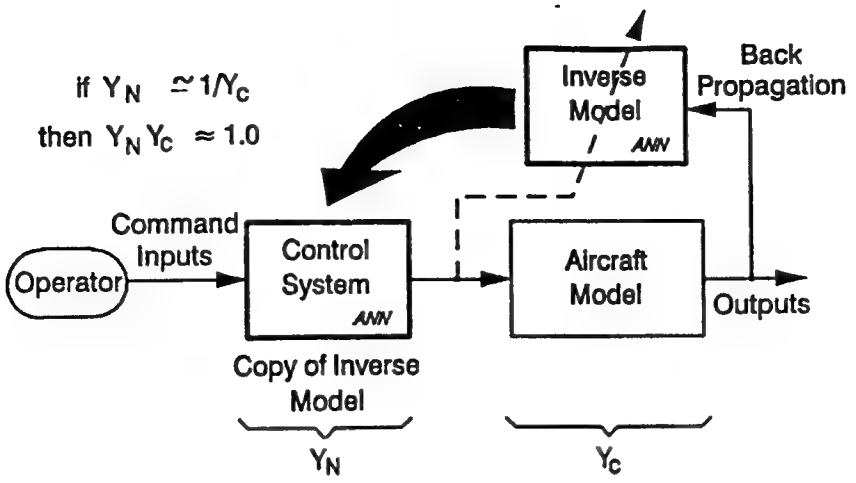
3. Kawato, M., Furukawa, K., and Suzuki, R., "A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement," Biological Cybernetics, Vol. 57, 1987, pp. 169 - 185.

4. Stites, R. and R.V. Walters, "Predicting Industrial Defect Behavior using Neural Networks," Proc. of ANN Applications, May 1991.

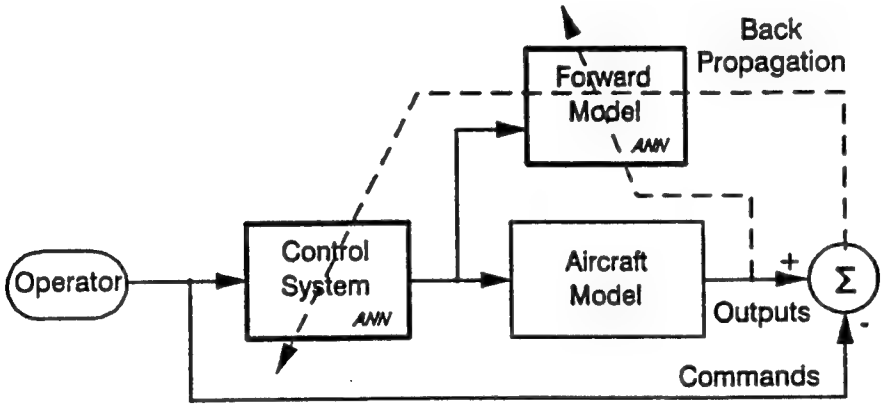
5. Walters, R.V., A Neural Model of Defect Behavior in a Complex Chemical Process, (unpub. Technical Report STR), 1989.

6. Jex, H.R., "On Improving the Detectability of Aircraft Pitch Control Impairment," Systems Technology, Inc., WP-1285-1, 1991.

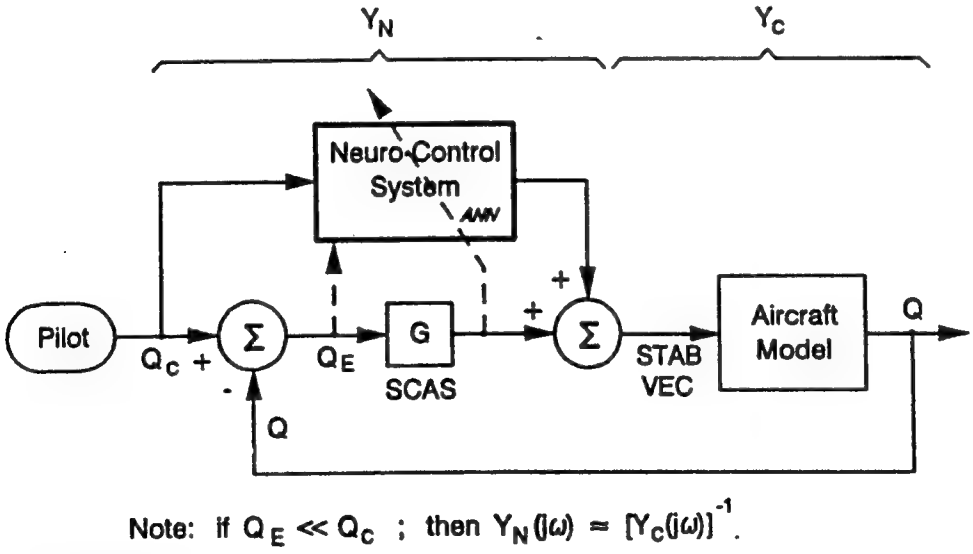
7. Jordan, M.I., and Jacobs, "Learning to Control an Unstable System with Forward Modeling," IEEE Conference on Neural Information Processing, Natural and Synthetic, 1989.



a) Direct Inverse Controller



b) Forward Model Controller



Note: if $Q_E \ll Q_C$; then $Y_N(j\omega) \approx [Y_C(j\omega)]^{-1}$.

c) Feedback Error Controller

Figure 1. Candidate Neurocontroller Architectures

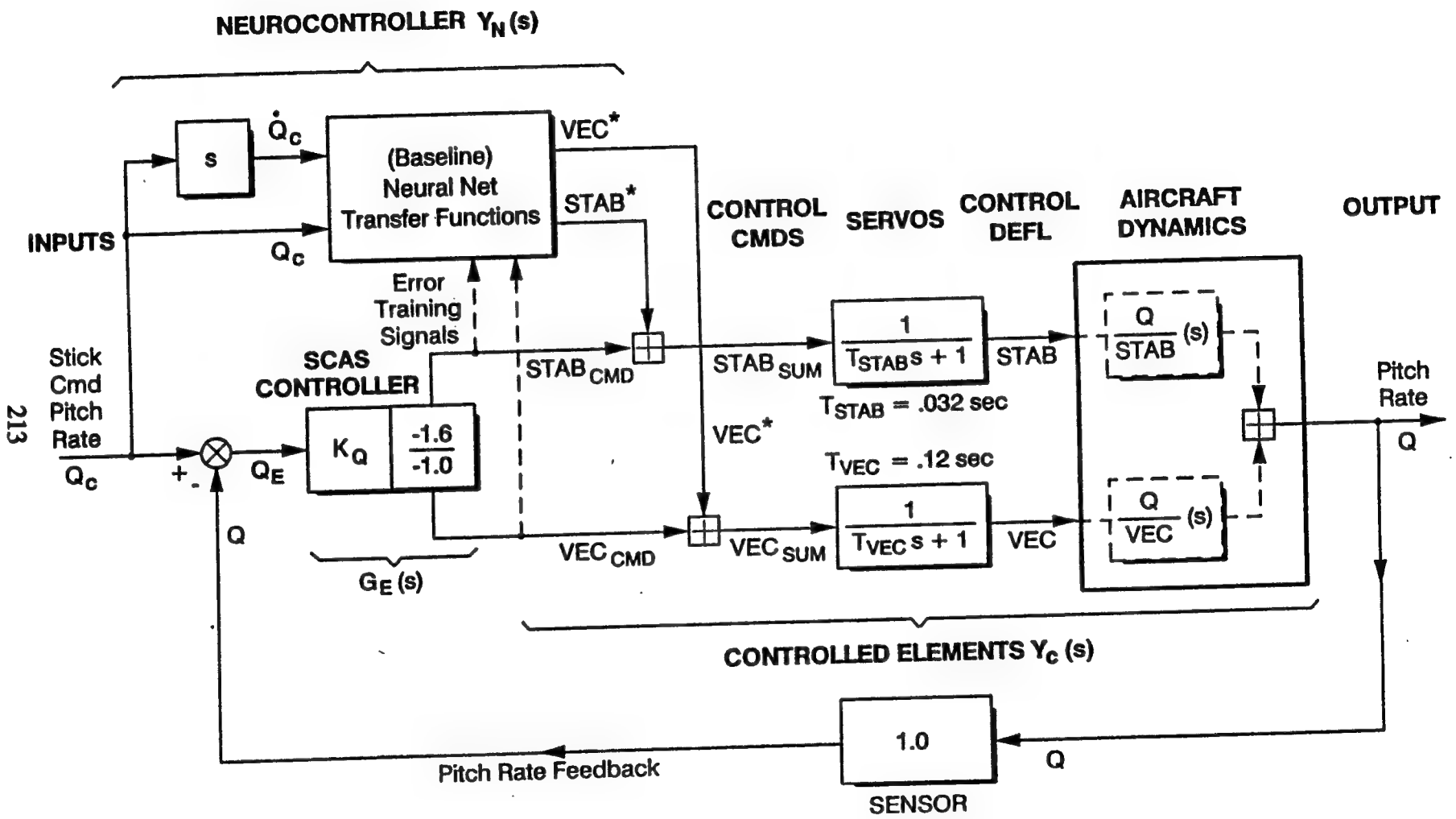


Figure 2. System Block Diagram

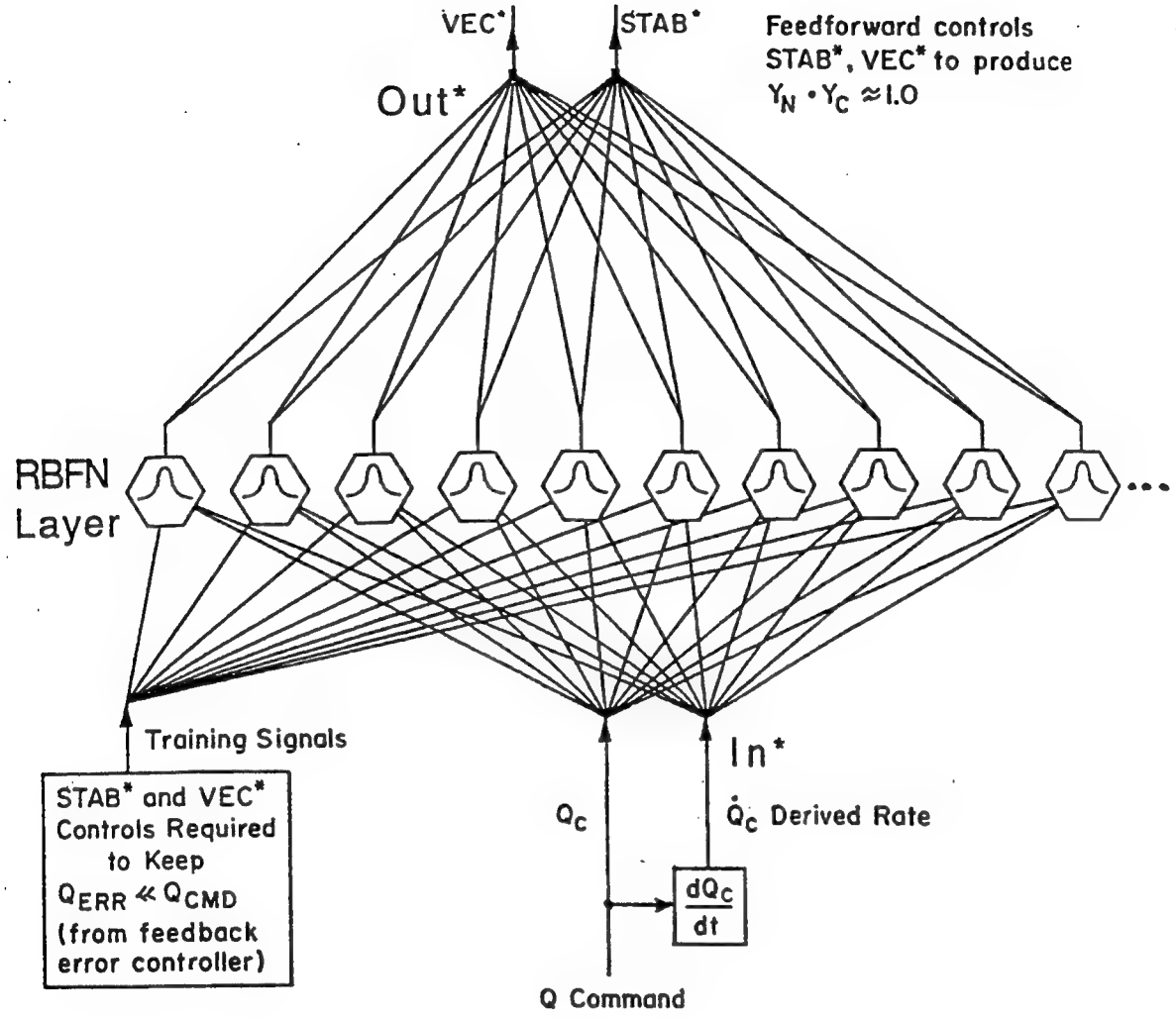


Figure 3. Baseline Configuration of Radial-Basis Function Neurons (STR Corp's Algorithm for Adaptive Clustering Network Used Hidden Layer)

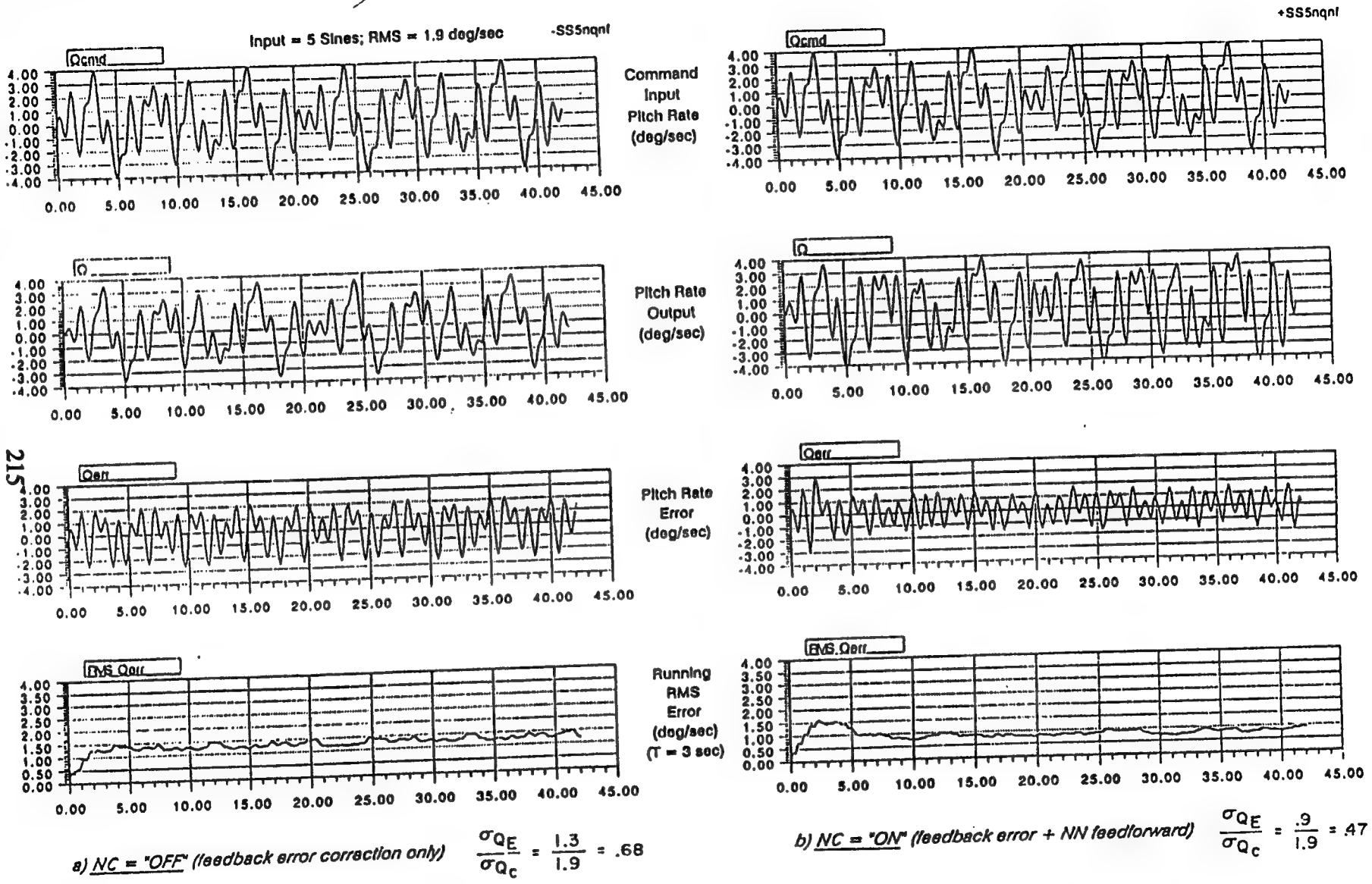


Figure 4. Effects of Neural Net Operation on Pitch Rate Signals
(Baseline Case, No Failures)

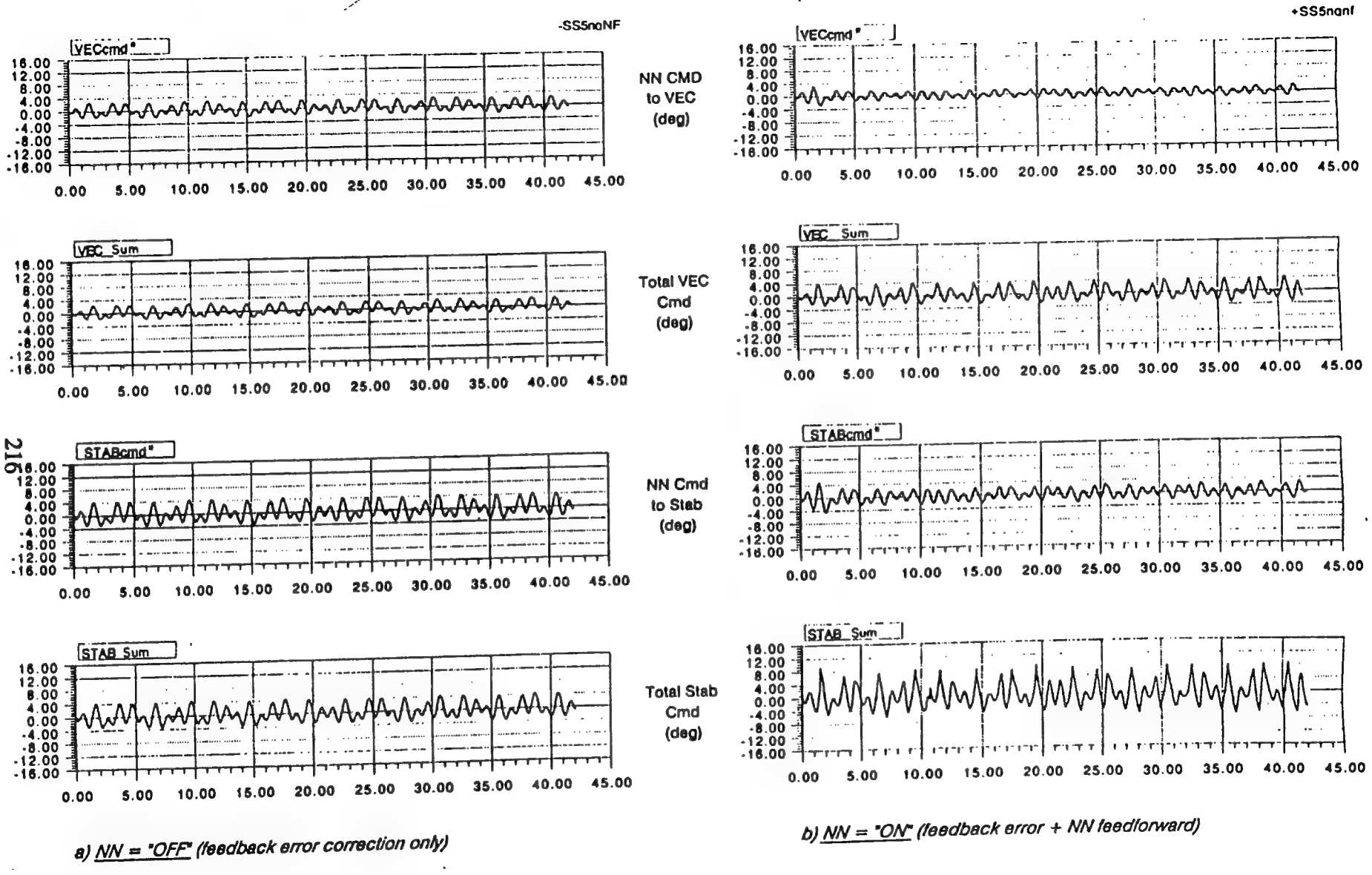
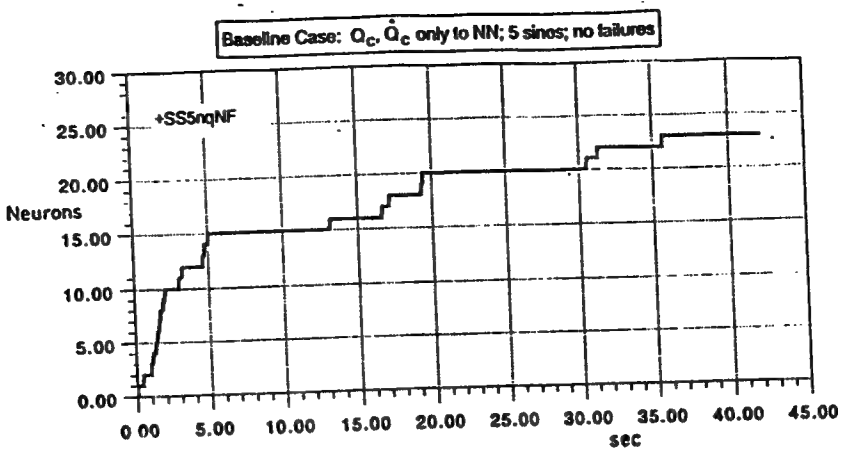
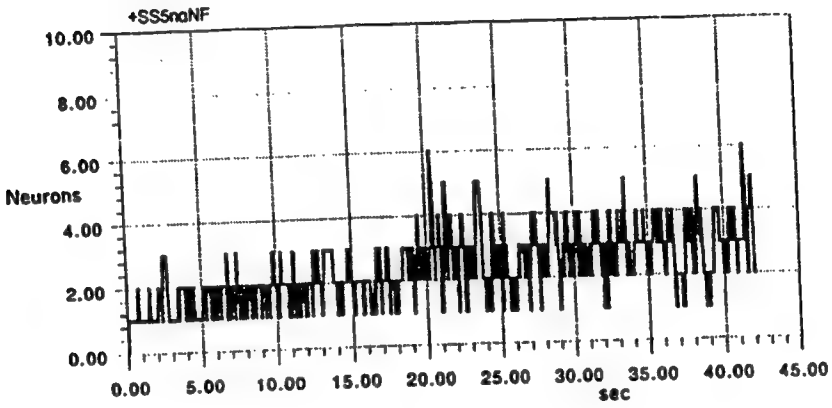


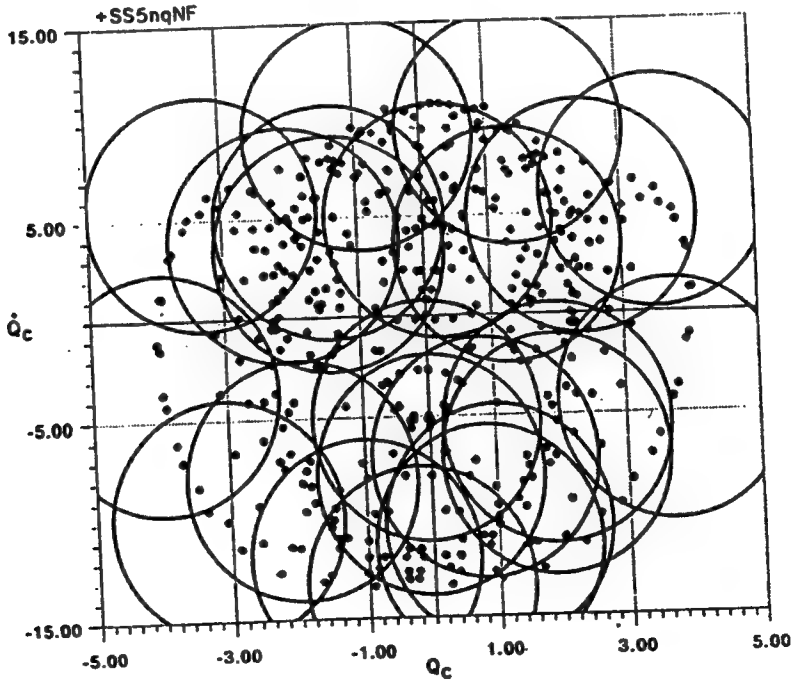
Figure 5. Effects of Neural Net Operation on Control Signals
(Baseline Case, No Failures)



a) Cumulative RBF Cells



b) Concurrently Active Cells



c) Centroids of ACN Cells After 21 sec

Figure 6. Neural Net Activity Measures
(Baseline Case)

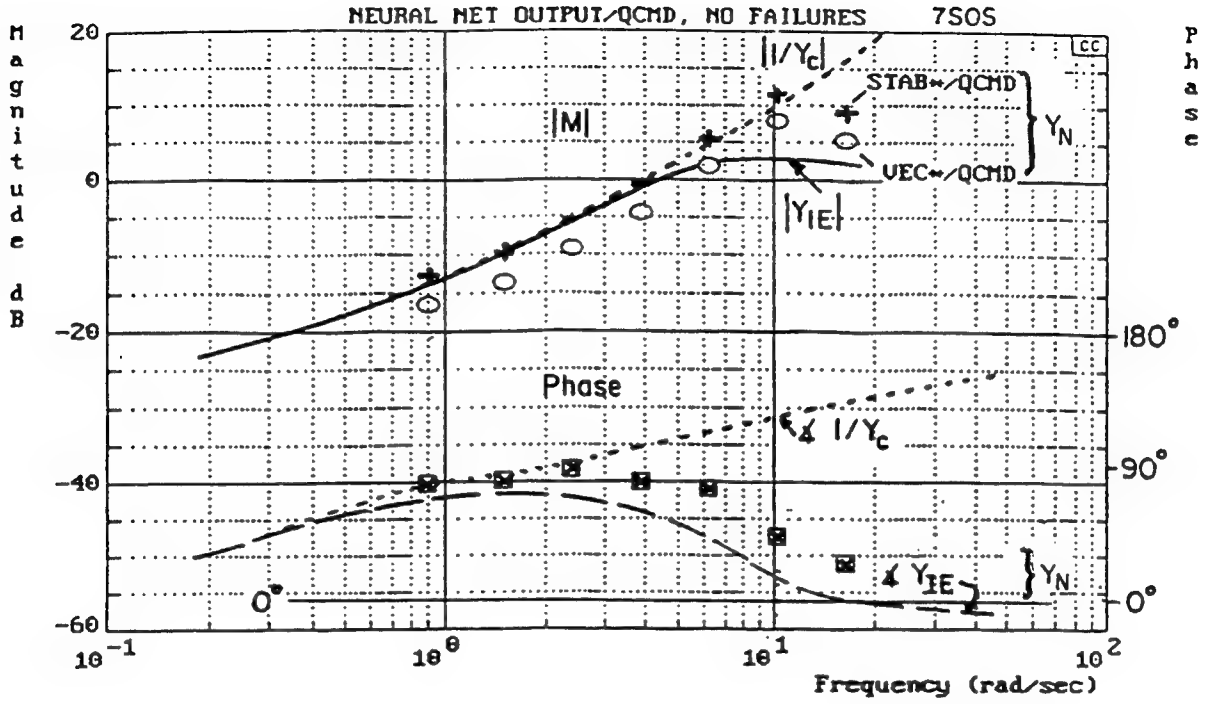
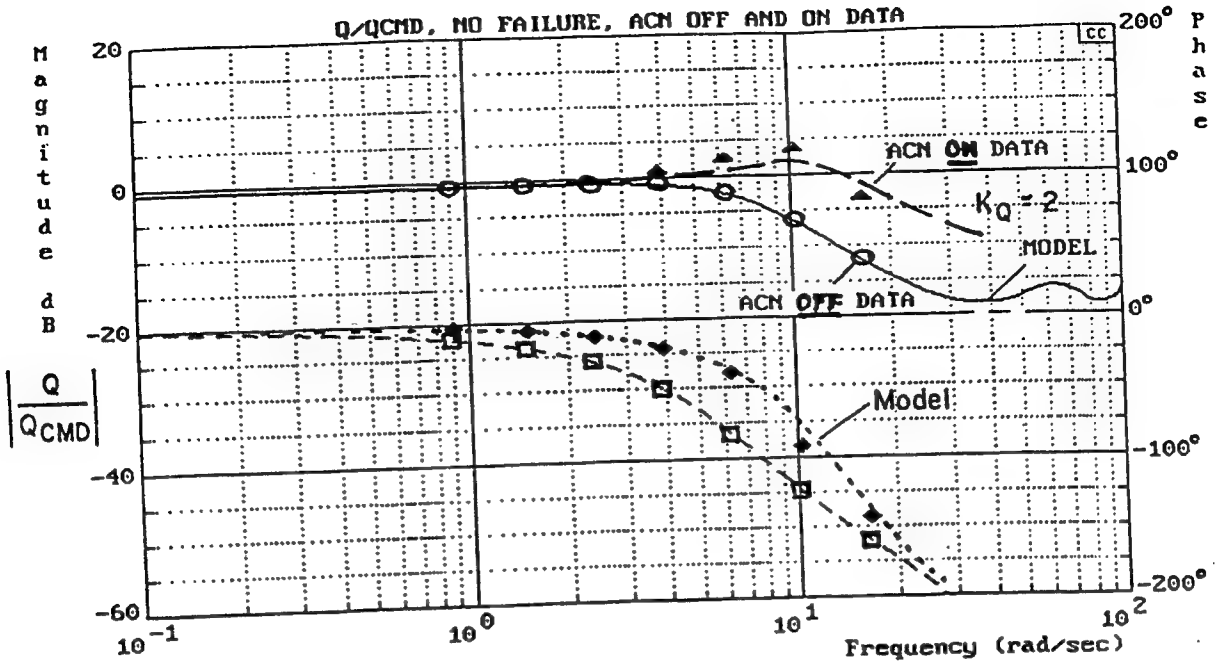
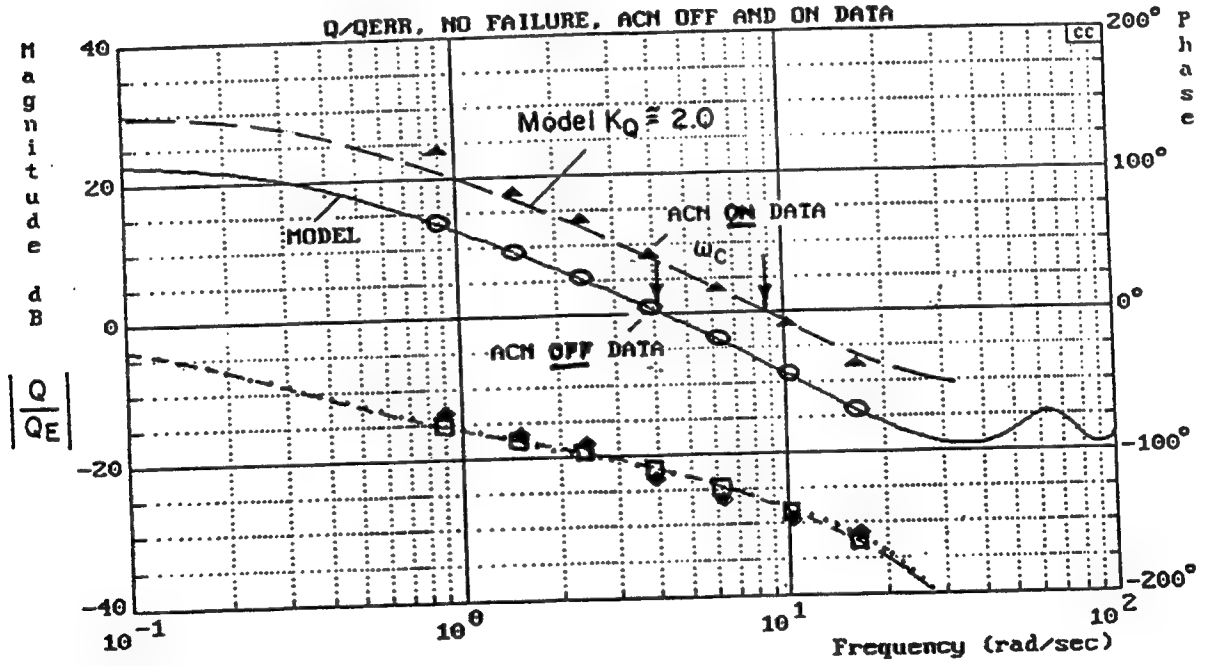


Figure 7. Comparison of Ideal and Measured Feedforward Describing Functions



a) Closed-Loop Q/Q_{CMD} Responses, NC OFF and ON



b) Equivalent Opened-Loop Q/Q_{ERR} Responses

Figure 8. Effect of Neurocontroller on Closed and Open-Loop Describing Functions

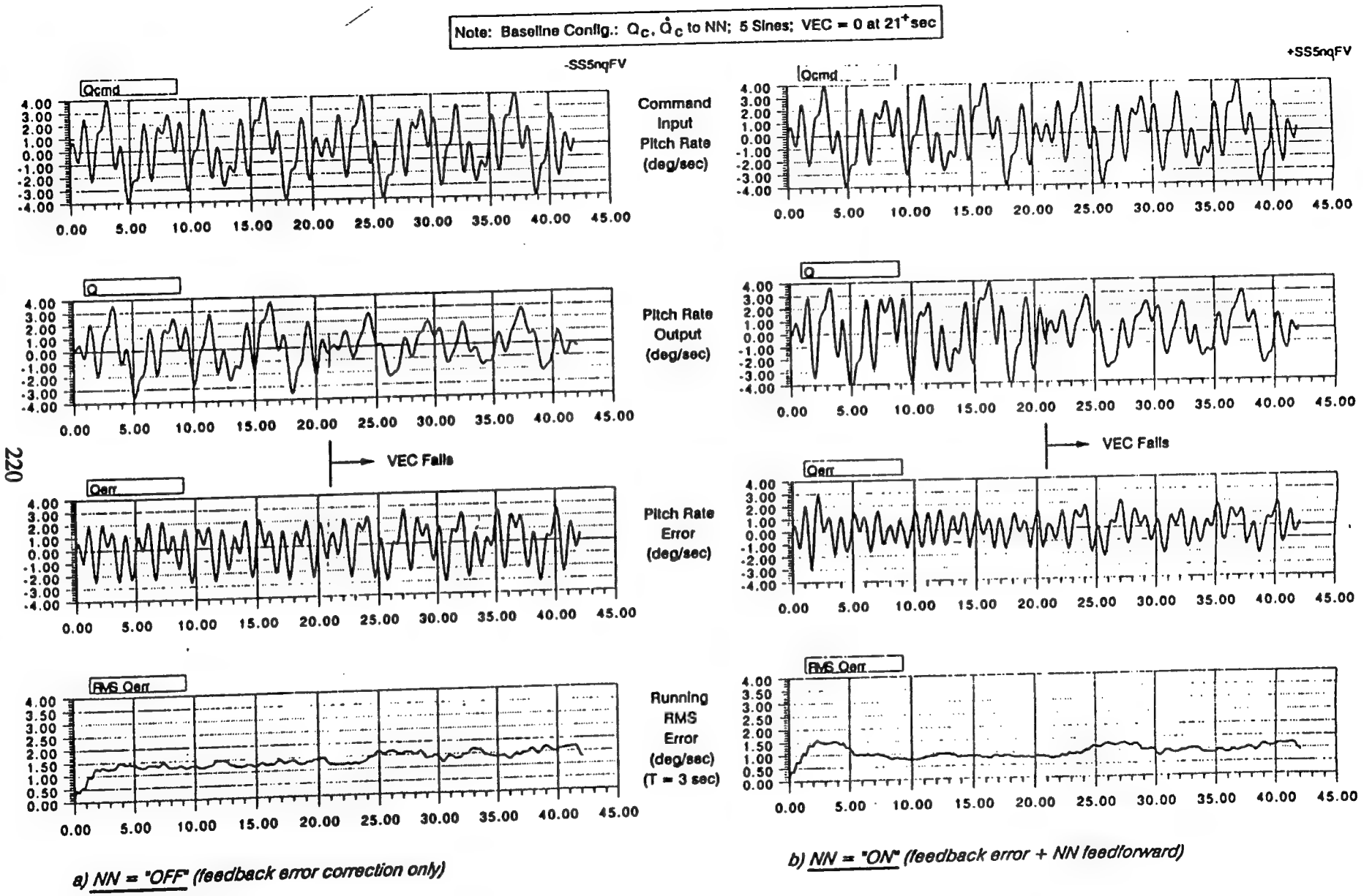


Figure 9. Effects of Thrust Vector Failure on Pitch Rates

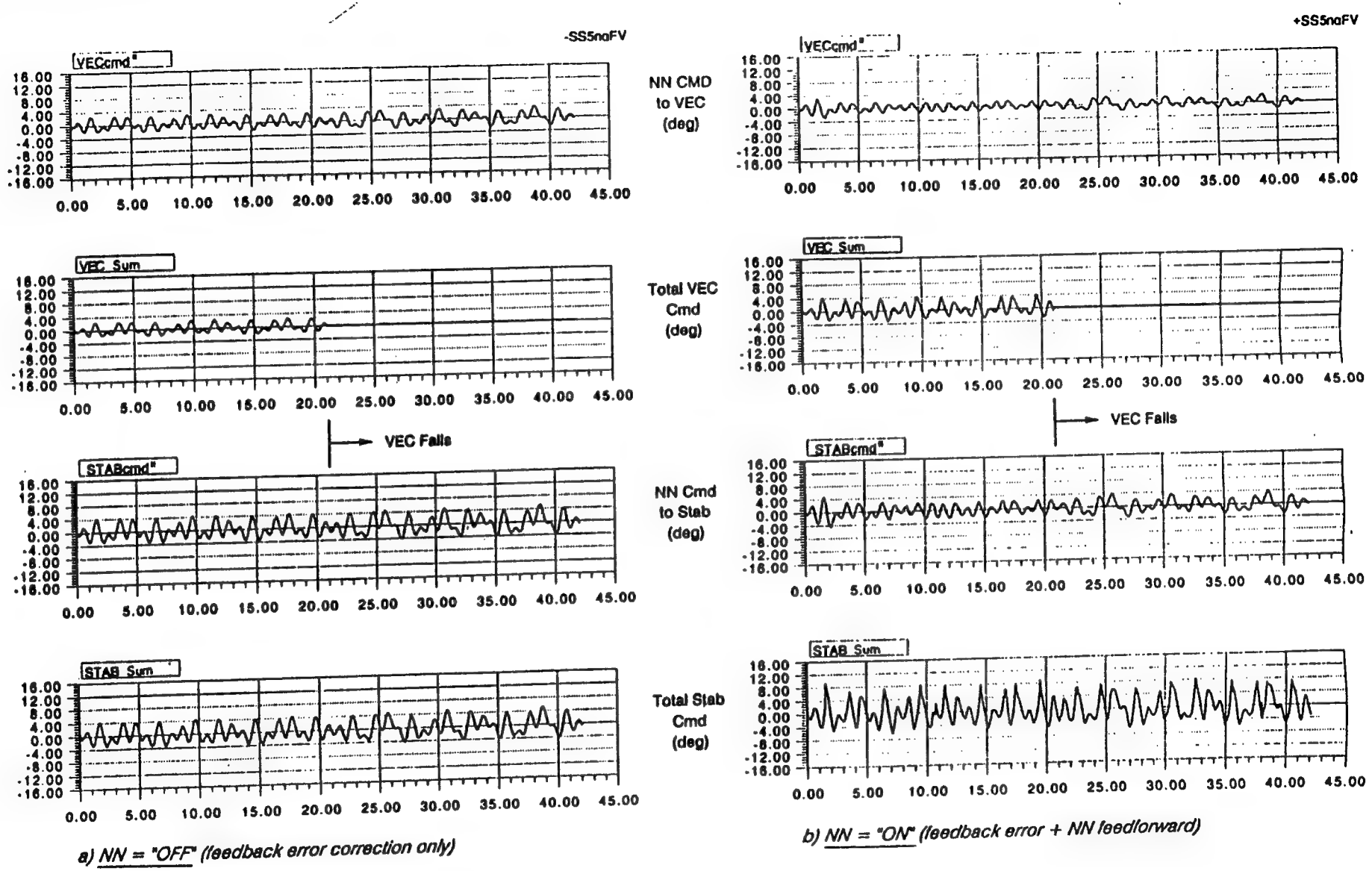


Figure 10. Effects of Thrust Vector Failure on Control Signals (VEC Fails @ 21^{+} Seconds)

Autonomous Neural Network Controller for Adaptive Material Handling

Dr. James Kottas
Dr. Michael Kuperstein
Symbus Technology, Inc.

Abstract: Current methods in motor control have problems dealing effectively with highly variable dynamic inertial interactions between multijointed robots and payloads. We are developing an autonomous neural network controller that can overcome these difficulties by learning to anticipate the inertial interactions from its own experience. The neural network controller will allow robots to handle diverse payloads in uncertain environments to benefit a wide variety of material handling applications. Our target application is bin-picking, the grasping of an object from a bin containing many randomly oriented objects and placing it at a desired location.

Our initial work has focused on the dynamic control aspect of the problem. Using a commercially available scara-type robot, we demonstrated a functional prototype of the neural network controller for realizing point-to-point control. The controller design consists of dynamic position and velocity servos in parallel with an adaptive neural network controller for each joint. The controller adaptively learns to compensate for the dynamic inertial interactions with different payloads through its own experience. Using two joints of the scara robot, the controller achieved a position accuracy of 0.2% of the joint range, a timing accuracy of within 5% of the requested movement time, and an end-point stability of within 8% of the maximum planned velocity. This performance was measured on both joints after only 150 training iterations with a movement that had large dynamic coupling forces between the scara links.

****Final Paper Not Submitted***

Other

A Neural Network Solution To The Real-Time Allocation of Marine Corps Tactical and C^3I Assets

Robert L. Stites
IKONIX Inc

"Nature is pragmatic rather than idealistic, demanding only that the goal path ... be tolerably suboptimal so that life-supporting needs are met and life-threatening situations avoided." P.N.Kugler

The efficiency with which assets are allocated during combat has usually played a dominant role in the outcome of the conflict _ often eclipsing all other factors, including variances in force structure. The solution of optimal allocation problems is no less critical to the design and operation of C^3I networks. Thus, the ability to perform this enabling task as efficiently as possible is vital.

The feasibility of using a hybrid artificial neural network (ANN) architecture to determine optimal allocations of a command's combat assets subject to a given set of constraints is illustrated. Utilizing as the case study an S-3 task that is at once extremely repetitive, complex, and important, the allocation of air assets to missions for the upcoming day, an enabling technology, and one which has a variety of easily seen C3I applications is demonstrated. The methodology for the integration of tactical information such as friendly and enemy force strengths and dispositions is described.

Inspiration and insight for a framework may be found in the behavioral, cognitive, neurophysiological, and engineering sciences. Allocation support and decision systems may be best described as an inference to the best explanation of the data and constraints. Problems in this domain are frequently described in terms of a constrained, multi-objective optimization model. The outcome of such optimization models is primarily due to relationships which exist between alternatives rather than principally due to properties of isolated data. It is, however, notable that the simple maximization, or minimization of a utility function is an insufficient model for real decision making. In any goal directed behavior of this nature, motivational influences must be included in the evaluation of choices. Selective attention is not sufficient to account for all of the motivational effects. The mathematical representation in a neural network must include, not only allowances for these motivational influences, but also an ability to accommodate soft, or ambiguous knowledge. A decision making framework supporting intentional and interdependent goals must be both path directed and goal motivated.

The heuristics employed in network generation renders a resulting artificial neural network which is more robust with respect to local minima and initial condition specification than other reported designs. Better qualitative results are achieved through the presence of a motivational bias or expectant objective. For large scale problems, an artificial neural network algorithm can be more computationaly efficient than traditional alternatives, even on serial digital processors.

Considerations of previously observed and published criticisms of artificial neural network optimization approaches are presented.

******Final Paper Not Submitted***

Noise Reduction System for Shipboard Spaces

Shinmin Wang and Robert Hecht-Nielsen
HNC, Inc.
5501 Oberlin Drive
San Diego, California 92121-1718

1.0 Introduction

Noise in shipboard spaces has been a problem for centuries. Noise makes working difficult, and hampers communications among crewmembers and between crew-members and the bridge. At the current epoch, no satisfactory solutions to the problems of noise suppression and free crew communication have been developed. Solutions to these problems would allow naval forces to operate more effectively, as well as improve the retention rate of the crews who must routinely work under such conditions. Currently, HNC is working on a Navy Phase II SBIR project[1], entitled *"Noise Reduction System for Shipboard Spaces"* (NRSS), which has as its goal the development of effective noise reduction and audio communication for crewmembers working aboard Navy ships. The primary technical objective of this Phase II SBIR is to build, and demonstrate aboard Navy ships, one brassboard and nine prototype NRSSs which are able to achieve the following performance measures:

- Total noise suppression for the variety of noise found in Navy shipboard spaces to a level well below the upper limit of 86 Db set by the U.S. Department of Labor for 8 hours of continuous sound exposure per day.
- The ability to communicate between crewmembers in the same airspace that are within 10 meters of each other and not badly shadowed.
- The ability of a crewmember to communicate with a bridge squawk box located not more than 10 meters away and not badly shadowed.
- The ability to communicate between crewmembers who are close (less than 2 meters) while speaking softly, and the inability of distant crew-members (those more than 5 meters away) to pick up these conversations.
- The ability of the average user to psychoacoustically determine the direction of another crewmember or the squawk box (to within ±30° in azimuth) from the reconstructed and audio imaged sound of their voice.
- A physical configuration suitable for widespread use in the Navy; i.e., comfortable to wear for continuous 4-hour duty shifts.

1

This project is sponsored by the Office of Naval Technology and managed by the David Taylor Research Center under contract #N00167-91-C-0060.

The brassboard system (which has been built and successfully demonstrated at sea see Section 3) was designed to demonstrate the system's functionality. The prototypes will be battery-powered portable units.

2.0 System Overview

The NRSS system has two functions. The first function is to reduce the intensity of environmental noise, and the second function is to provide communications capability.

2.1 Noise Attenuation Subsystem

The noise reduction function is carried out via two mechanisms: passive attenuation by physically blocking the noise entering each ear, and active attenuation carried out by an information processing device utilizing a microphone/speaker set at each ear. Foam ear plugs are the most effective means of passive sound attenuation developed to date. The brand of foam ear plugs chosen for the NRSS (EAR plugs by Cabot Corporation) attenuate sound power levels in the frequency range 20 Hz - 20,000 Hz by 24 dB or more. Almost all of the loud noises found in Navy ships lie in the frequency range between 20 Hz and 1,000 Hz, with the majority of the sound power typically being found below 300 Hz. Unfortunately, it is in this lowest frequency range that the ear plugs have their lowest passive noise attenuation performance. Thus, we need to augment the ear plugs with an additional noise cancellation mechanism.

The active noise cancellation technique used in NRSS employs a microphone/speaker set at each ear. Any noise picked up by the microphone is electronically converted to a signal and passed through a special linear stabilizing loop filter. The output of the noise cancellation loop filter is then used to drive the ear speaker. In the NRSS design, both the microphone and the speaker are placed within the ear canal on the inside of the ear plug (see Figure 1). The miniature speaker and microphone components are commercially available and will be shrouded in a soft material to ensure complete wearer comfort when used with the foam ear plugs. The microphone and speaker assembly and the foam ear plugs will be easily removable for cleaning so that multiple personnel can use the sets.

To provide additional cancellation, we have experimented with adding a neural network canceler to the basic loop filter canceler. This neural network (a recurrent backpropagation neural network - see [1]) is trained using a desired speech signal mixed with recorded ship noise. The input to the network is the residual signal after active cancellation by the linear loop filter. The desired output of the network is the desired speech signal with all noise canceled. Experiments have shown that this recurrent network can provide additional cancellation over a linear loop filter; however, implementing the network is computationally expensive.

2.2 Communications Subsystem

The other part of the NRSS system is designed to support free and natural communications

among crewmembers, and between crewmembers and the bridge. The operation of this part of the system is now described.

The communications system operates via an ultrasonic transmitter and receiver array assembly mounted on top of the crewmember's head (see Figure 1). The array is attached with Velcro and can be mounted on a helmet. A noise canceling boom microphone (also attached via Velcro) is used to pick up the crewmember's voice. The microphone is always "hot"; i.e., there are no buttons to push or speech activated switches to operate - the system is always on in both the transmit and receive modes.
Whenever the crewmember speaks, the sound picked up by the microphone is upconverted to 40 kHz by mixing with a local oscillator within the processor/powerpack. This signal is then transmitted by the ultrasonic transmitter. The transmitter, which fires upward, has a conical deflector above it which directs the sound in an omni-directional pattern. Each receiver array contains 8 directional ultrasound receivers that have elliptical directivity patterns (with the ellipse about 60° wide in the horizontal plane, and about 120° wide in a vertical plane). These receivers are spaced unevenly around the 360° azimuth circle to reflect object discrimination patterns of the human hearing system. The signal from each of the 8 ultrasonic receivers is continuously downloaded to an audio band signal by envelop detection, low-pass filtering, and amplification. Each downconverted signal is then passed through a head-related transfer function filter that approximates the signal the eardrum would hear from a sound arriving from that direction due to the filtering caused by the head and ear pinna.

The system is full duplex and multiple crewmembers can speak at once, if desired. However, just as with ordinary speech, if multiple people speak at the same time they interfere with each other (unless they are far away from one another or are speaking quietly).

3.0 System Design

The processor/power pack of the prototype system contains an analog circuit board, a digital circuit board, and a battery pack. The audio circuits and the A/D and D/A interface circuits reside on the analog circuit board. Figure 2 has a functional block diagram for the NRSS analog board. The audio circuits are for one transmit channel and eight receive channels for voice communications and audio imaging. The A/D interface contains the A/D function for the eight voice channels and two ear microphone channels. The D/A interface contains the D/A function for the left and right ear speaker channels. The A/D and D/A interface circuits are controlled by the digital signal processor on the digital circuit board via the control logic circuits.

The digital circuit board contains the microprocessor (TMS320C25), the 40 Mhz clock, and the startup circuit for the TMS320C25. The signal processing software is executed on the TMS320C25 to provide the audio imaging and the active noise cancellation functions. Figure 3 shows the block diagram for the NRSS digital circuit board.

The battery pack contains the batteries needed to power the system during operational testing.

4.0 Project Progress

The NRSS project is currently a Phase II Navy SBIR. A brassboard version of the system was built and successfully demonstrated aboard the Navy SES-200 ship in April 1992. In the NRSS brassboard, a data acquisition board (A/D, D/A) and a signal processing board were added to an IBM PC compatible microcomputer to perform the functions of the A/D and D/A interface board and the digital board.

During the demonstration of the NRSS brassboard, we demonstrated the voice communication function in an extremely noisy engineering space (95 to 118 dB noise level). Intelligibility of voice communication between crewmembers was good for ranges over 39 feet. The brassboard system audio imaging worked fair in a quiet environment, but performance in loud noise was poor. It is believed that after some tuning of the system parameters to compensate for unevenness in receiver sensitivities and other physical components, the full audio imaging effect will be achieved. The imaging filters (which were developed using data provided by the U.S. Air Force Bioacoustics Lab) have been tested separately and they provide excellent azimuth discrimination.

The NRSS active noise cancellation function was tested subjectively by the crewmembers. Measurements of the active noise cancellation function performance were made by using a KEMAR head. The analysis of these measurements is still in progress. Prototypes of the NRSS system with the processor/power pack as described above are now being built and will be delivered to the project office later in Phase II for Navy experiments.

5.0 Benefits and Future Applications

Our goal is to develop an NRSS system that will be an effective, comfortable, and lightweight hearing protection and communications system for shipboard personnel. In Phase III we anticipate developing a production model of the system that will be suitable for use on Naval and Coast Guard vessels, submarines, Army and Marine Corps tanks, and on Air Force flight lines, as well as in civilian applications.

6.0 References

[1] Robert Hecht-Nielsen, **Neurocomputing**, Addison-Wesley, 1991.

NOISE REDUCTION SYSTEM FOR SHIPBOARD SPACES – PHASE II

PROTOTYPE SYSTEM OVERVIEW

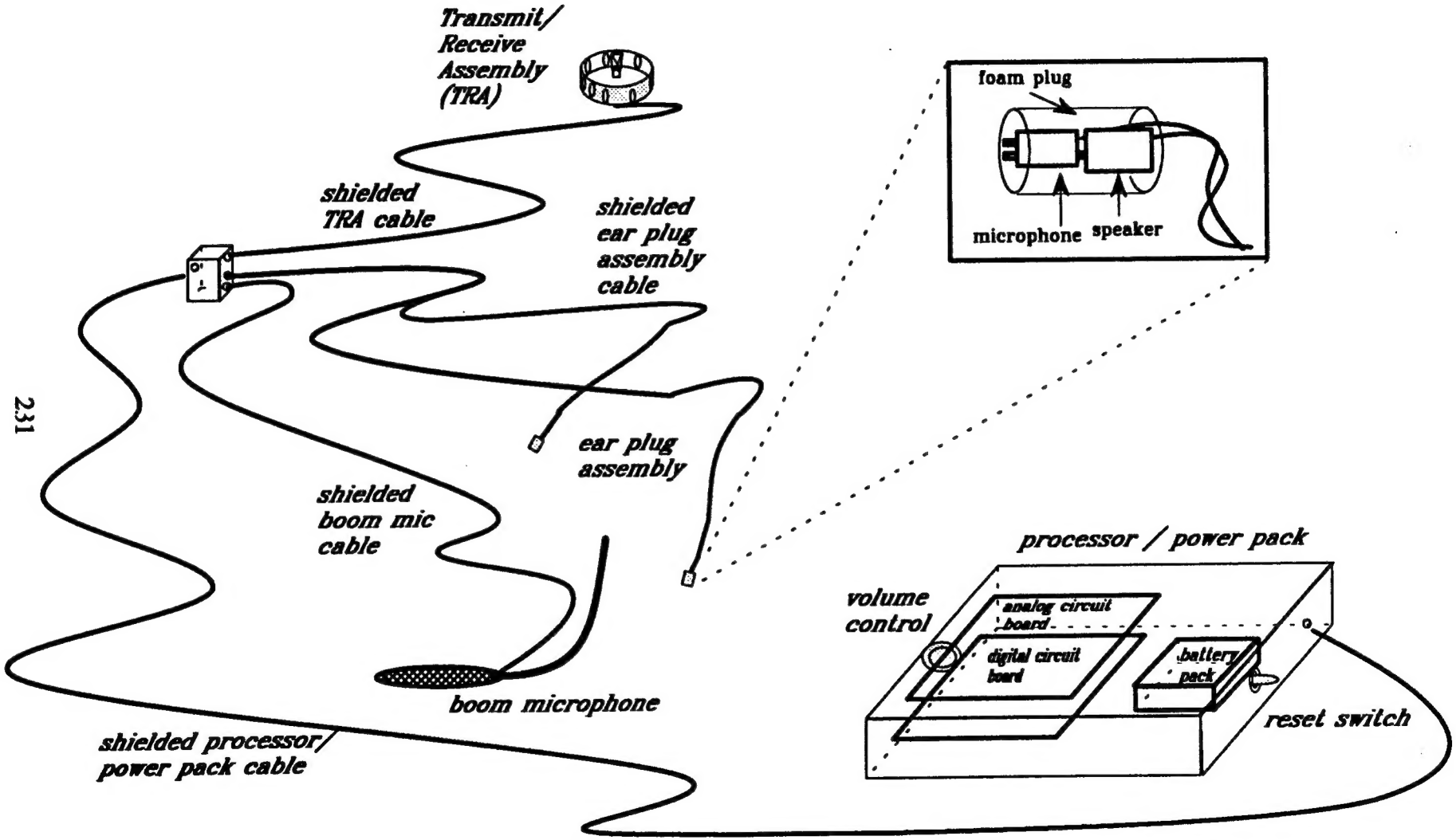
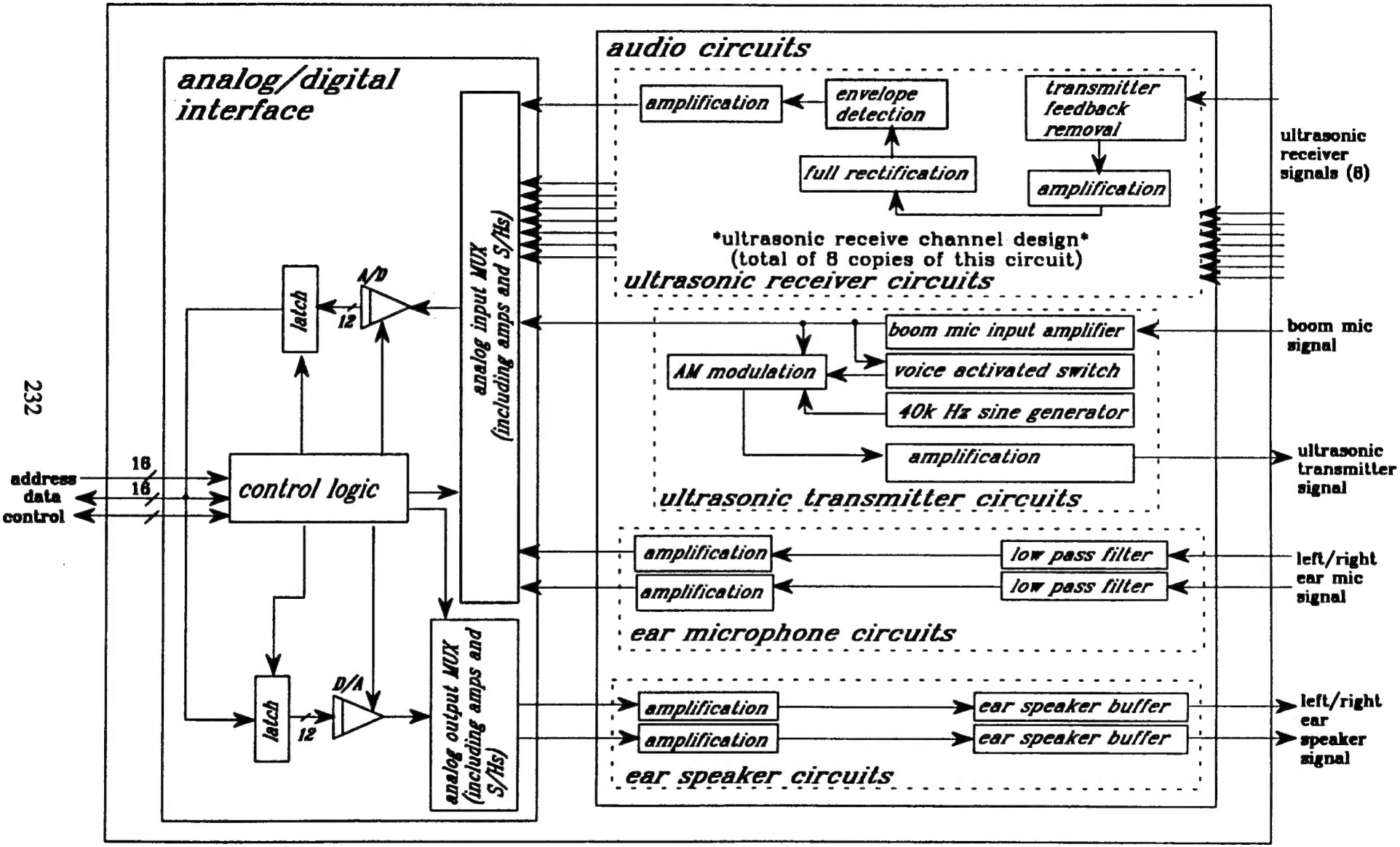


Figure 1

NOISE REDUCTION SYSTEM FOR SHIPBOARD SPACES – PHASE II

PROTOTYPE SYSTEM ANALOG CIRCUIT BOARD



nrpaper2

Figure 2

NOISE REDUCTION SYSTEM FOR SHIPBOARD SPACES – PHASE II

PROTOTYPE SYSTEM DIGITAL CIRCUIT BOARD

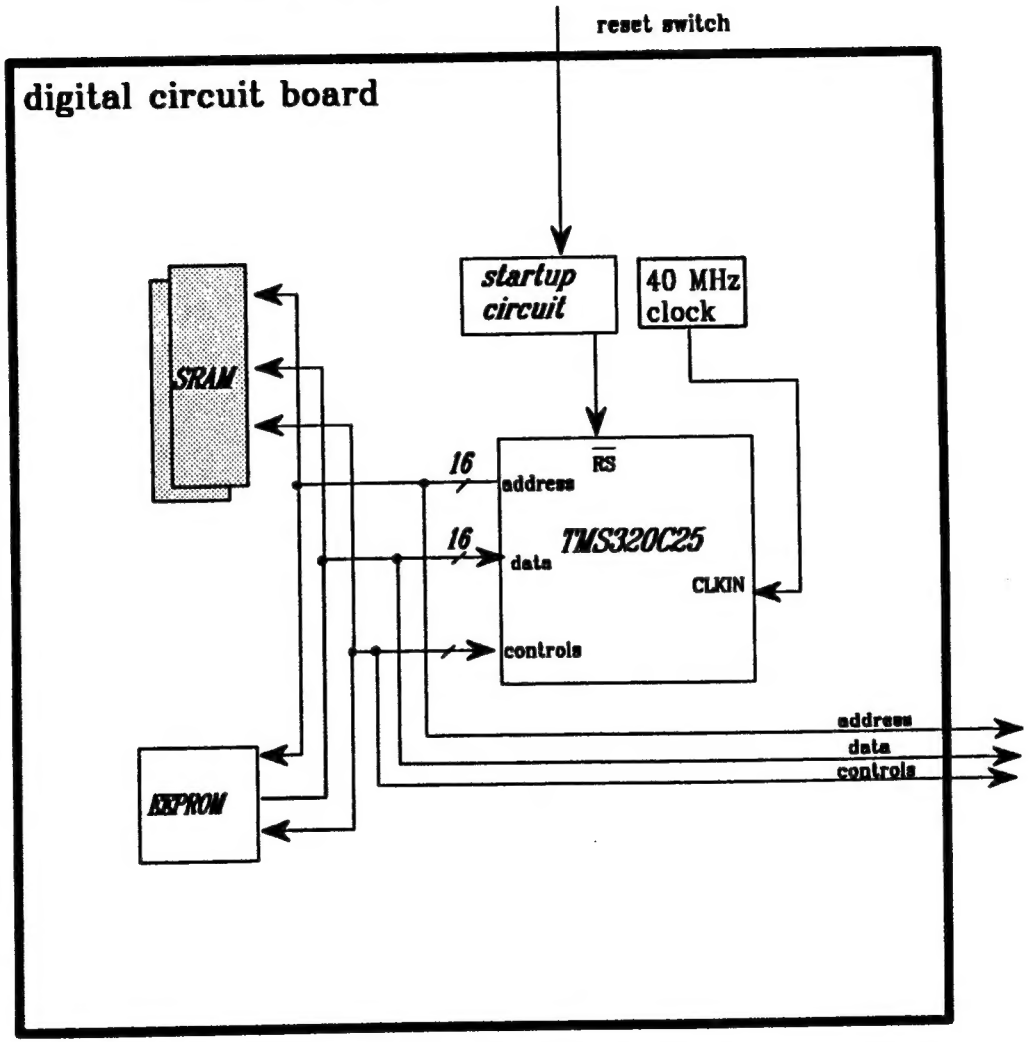


Figure 3